Pontifícia Universidade Católica
do Rio de Janeiro

Luís Marcelo Vital Abreu Fonseca

Classificação de Objetos em Contexto Real por Redes Neurais Convolutivas

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio.

Orientador: Prof. Ruy Luiz Milidiú

Rio de Janeiro
Março de 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Luís Marcelo Vital Abreu Fonseca

Classificação de Objetos em Contexto Real por Redes Neurais Convolutivas

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Ruy Luiz Milidiú
Orientador
Departamento de Informática – PUC-Rio

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Prof. Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Prof. Marcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 07 de Março de 2017

Luís Marcelo Vital Abreu Fonseca

Graduou-se em Engenharia da Computação na PUC-Rio em 2015. Seus interesses incluem otimização de sistemas, inteligência artificial e processamento de imagens.

Ficha Catalográfica

Fonseca, Luís Marcelo Vital Abreu

Classificação de Objetos em Contexto Real por Redes Neurais Convolutivas / Luís Marcelo Vital Abreu Fonseca; orientador: Ruy Luiz Milidiú. – 2017.

56 f. : il. color. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2017.

Inclui bibliografia

1. Informática – Teses. 2. Aprendizado de Máquina. 3. Processamento de Imagens. 4. Redes Neurais Convolutivas. 5. Classificação de Objetos. 6. Mineração de Dados. I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Agradeço ao meu orientador Ruy Luiz Milidiú pela parceria e dedicação para a realização deste trabalho.

A Capes e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus pais, pela educação e confiança de todas as horas.

Aos meus colegas da PUC-Rio.

Aos professores e funcionários do Departamento de Informática.

Resumo

Fonseca, Luís Marcelo Vital Abreu; Milidiú, Ruy Luiz. **Classificação de Objetos em Contexto Real por Redes Neurais Convolutivas**. Rio de Janeiro, 2017. 56p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A classificação de imagens em contexto real é o ápice tecnológico do reconhecimento de objetos. Esse tipo de classificação é complexo, contendo diversos problemas de visão computacional em abundância. Este projeto propõe solucionar esse tipo de classificação através do uso do conhecimento no aprendizado de máquina aplicado ao *dataset* do MS COCO. O algoritmo implementado neste projeto consiste de um modelo de Rede Neural Convolutiva que consegue aprender características dos objetos e realizar predições sobre suas classes. São elaborados alguns experimentos que comparam diferentes resultados de predições a partir de diferentes técnicas de aprendizado. É também realizada uma comparação dos resultados da implementação com o estado da arte na segmentação de objetos em contexto.

Palavras-chave

Aprendizado de Máquina; Processamento de Imagens; Redes Neurais Convolutivas; Classificação de Objetos; Mineração de Dados;

Abstract

Fonseca, Luís Marcelo Vital Abreu; Milidiú, Ruy Luiz (Advisor). **Classification of Objects in Real Context by Convolutional Neural Networks**. Rio de Janeiro, 2017. 56p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The classification of objects in real contexts is the technological apex of object recognition. This type of classification is complex, containing diverse computer vision problems in abundance. This project proposes to solve that type of classification through the use of machine learning knowledge applied to the MS COCO dataset. The implemented algorithm in this project consists of a Convolutional Neural Network model that is able to learn characteristics of the objects and predict their classes. Some experiments are made that compare different results of predictions using different techniques of learning. There is also a comparison of the results from the implementation with state of art in contextual objects segmentation.

Keywords

Machine Learning; Image Processing; Convolutional Neural Networks; Object Classification; Data Mining;

Sumário

1 Introdução **14**
1.1 Classificação e Segmentação 15
1.2 Motivação e Objetivo 16
1.3 Dataset 16
1.4 Estado da Arte 17
1.5 Estrutura da Dissertação 19

2 Conceitos **22**
2.1 Redes Neurais 22
2.2 Feed-Forward 23
2.3 Função de Ativação 23
2.4 Backpropagation 24
2.5 Hiperparâmetro 25
2.6 Overfitting 25
2.7 Instância 25
2.8 Métricas 26
2.9 Redes Neurais Convolutivas 26
2.9.1 Camada Convolutiva 28
2.9.2 Zero-Padding 28
2.9.3 Dropout 29
2.9.4 Camada ReLU 29
2.9.5 Camada de Pooling 29
2.9.6 Camada Densa 30
2.10 Redes Neurais Residuais 30
2.11 Pré-Processamento 30
2.11.1 Redimensionamento 30
2.11.2 Flip 31
2.11.3 Shift 31

3 Metodologia e Resultados **34**
3.1 Arquitetura 1 - CNN 35
3.2 Arquitetura 2 - ResNet 35
3.3 Pré-Processamento e Aumento do Dataset 35
3.4 Testes Automatizados 36
3.5 Experimento 1 - Comparação: CIFAR-10 vs COCO 37
3.5.1 Descrição 37
3.5.2 Resultados 37
3.6 Experimento 2 - Aumento de dataset e padding de instâncias pequenas 38
3.6.1 Descrição 38
3.6.2 Resultados 38
3.7 Experimento 3 - Comparação: CNN vs ResNet 39
3.7.1 Descrição 39
3.7.2 Resultados 39
3.8 Experimento 4 - Comparação: Bounding Box vs Máscara de Segmentação 40

3.8.1 Descrição 40
3.8.2 Resultados 40
3.9 Experimento 5 - Resultados Finais 40
3.9.1 Descrição 40
3.9.2 Resultados 41
3.10 Comentários 43

4 Conclusões e Trabalhos Futuros **51**

Referências bibliográficas **54**

Lista de figuras

Figura 1 Sub-problemas da compreensão de imagens: (a) classificação dos objetos; (b) localização de *bounding boxes* de objetos; (c) segmentação semântica de *pixel*; e (d) segmentação e classificação de objetos individuais. 14

Figura 2 Captcha moderno que tenta dificultar o reconhecimento da imagem por sistemas automatizados através da distorção do texto e adição de uma linha que dificulta a segmentação dos caracteres 15

Figura 3 Exemplo de imagem de contexto real do banco MS COCO. Notam-se várias dificuldades para o reconhecimento dos objetos, como por exemplo, a baixa resolução das sub-imagens de pessoas por conta da distância para a câmera, o ângulo de perspectiva do ônibus, e a oclusão dos carros pelos prédios. 17

Figura 4 As diferenças entre imagens de contexto real do MS COCO e outros bancos de processamento de imagens. 18

Figura 5 Em cima, uma imagem com suas anotações de máscaras de segmentação. Abaixo, a mesma imagem com suas anotações de *bounding boxes*. 19

Figura 6 Diferença entre uma imagem com e sem seus rótulos de máscara de segmentação 20

Figura 7 Esquema da unidade McCullock - Pitts, modernamente conhecido como um neurônio artificial. 22

Figura 8 Uma Rede Neural Feed-Foward com 3 camadas totalmente conectadas. 23

Figura 9 Uma DNN convencional que possui suas camadas representadas por neurônios em 2 dimensões. 27

Figura 10 Uma CNN que possui suas camadas representadas por neurônios em 3 dimensões. 27

Figura 11 Um exemplo de cálculo do a de um neurônio. Para realizar o cálculo, basta multiplicar os valores das entradas com os valores nos filtros nas respectivas posições e realizar a soma de todos eles. Neste caso, $a = 2 \times 1 + 2 \times (-1) + 1 \times 1 + 1 \times (-1) + 1 \times (-1) = -1$. 29

Figura 12 A convolução de um filtro de tamanho $F = 3$ em uma entrada de dimensão $5 \times 5 \times 1$ com $S = 1$. Cada linha indica uma iteração da convolução. 32

Figura 13 Um exemplo de entrada com diferentes valores de Zero-Padding. 33

Figura 14 Um exemplo com duas iterações de um Max Pooling com $F = 2$ e $S = 2$. Cada cor diferente mostra um campo receptivo do pooling. 33

Figura 15 Uma camada residual é composta de duas camadas de convolução com uma função de ativação ReLU entre elas. A saída da camada residual é a função ReLU da sua própria entrada acrescentada da saída da segunda camada intermediária. 33

Figura 16 A arquitetura 1 apresenta uma CNN convencional com camadas convolutivas, de pooling, e densa. 45

Figura 17 A arquitetura 2 apresenta uma ResNet mais complexa com camadas convolutivas e residuais. 46

Figura 18 O pré-processamento realizado nas instâncias antes de serem alimentadas para a rede. Instâncias de tamanho maior que 32x32 são redimensionadas enquanto instâncias menores sofrem um *padding* de *pixels* pretos até 32x32. 47

Figura 19 A diferença em precisão dos resultados obtidos utilizando os rótulos de segmentação. 48

Figura 20 A diferença em *recall* dos resultados obtidos utilizando os rótulos de segmentação. 48

Figura 21 A diferença em tempo de execução dos resultados obtidos utilizando os rótulos de segmentação. 49

Figura 22 Comparação entre os resultados de precisão obtidos no experimento 5 e o estado da arte do grupo MSRA utilizando apenas 10 classes. 49

Figura 23 Comparação entre os resultados de precisão obtidos no experimento 5 e o estado da arte do grupo MSRA utilizando todas as classes do *dataset*. O ID da classe é ordenado conforme as posições das classes na tabela 7. Isso significa que o gráfico está em ordem decrescente de resultados de F_1 obtidos no experimento 5. As linhas em azul indicam diferença positiva do experimento com o estado da arte e as linhas em vermelho indicam diferença negativa. 50

Figura 24 A sobreposição de objetos pode fazer com que as anotações fiquem segmentadas, tornando o objeto um polígono descontínuo. 53

Lista de tabelas

Tabela 1 Lista de competidores do MS COCO Challenge 2015. 20

Tabela 2 Resultados do grupo MSRA no Challenge 2015 para a segmentação da *bounding box* em 10 classes diferentes escolhidas de forma arbitrária. 21

Tabela 3 Resultados do experimento 1 no *dataset* COCO em diferentes métricas. Resultados em ordem decrescente de F_1. 37

Tabela 4 Resultados do experimento 2 no *dataset* COCO em diferentes métricas. Resultados em ordem decrescente de F_1. 38

Tabela 5 Resultados do experimento 3 no *dataset* COCO em diferentes métricas. Resultados em ordem decrescente de F_1. 39

Tabela 6 Comparação entre a arquitetura 1 e a arquitetura 2, utilizando a média das métricas entre todas 10 classes. 39

Tabela 7 Desempenho da arquitetura 2 no *dataset* completo do MS COCO. Resultados em ordem decrescente de F_1. 43

Lista de Abreviaturas

NN – *Neural Network*
DNN – *Deep Neural Network*
MLP – *Multilayer Perceptron*
CNN – *Convolutional Neural Network*
ResNet – *Residual Neural Network*
ReLU – *Rectified Linear Unit*
MSE – *Mean Squared Error*
GPU – *Graphics Processing Unit*

People worry that computers will get too smart and take over the world, but the real problem is that computers are too stupid and they've already taken over.

Pedro Domingos, *The Master Algorithm.*

1
Introdução

Um dos principais objetivos da visão computacional é a compreensão de cenas visuais. Na figura 1, ilustramos a compreensão de uma cena que envolve inúmeras tarefas, incluindo o reconhecimento dos objetos que estão presentes, a localização dos objetos em 2D e 3D, a determinação dos objetos e dos atributos da cena, a caracterização das relações entre objetos e o fornecimento de uma descrição semântica da cena, em graus crescentes de complexidade.

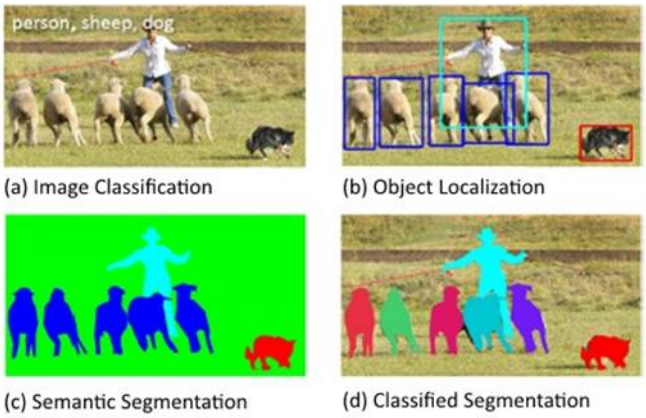

Figura 1: Sub-problemas da compreensão de imagens: (a) classificação dos objetos; (b) localização de *bounding boxes* de objetos; (c) segmentação semântica de *pixel*; e (d) segmentação e classificação de objetos individuais.

O avanço tecnológico da computabilidade (18) possibilitou o uso sistemático de modelos de alta complexidade matemática. As Redes Neurais são modelos capazes de aprender e reconhecer padrões em diversos problemas. Entretanto, as primeiras redes criadas eram incapazes de processar de forma eficiente tarefas relacionadas com o processamento de imagens por conta da grande quantidade de informação que cada imagem possui. Sendo assim, as tarefas de processamento de imagens eram realizadas principalmente a partir do conhecimento de domínio.

Porém, em 1998, as Redes Neurais Convolutivas alteraram esse cenário, possibilitando que o reconhecimento de imagens fosse feito utilizando técnicas de aprendizado de máquina. Isso modificou o paradigma da área, fazendo com

que o conhecimento de domínio para o processamento de imagens fosse menos necessário.

Além de afetar os profissionais da área, essa revolução tecnológica abriu novas possibilidades que antes eram inalcançáveis. Um exemplo disto é ilustrado na figura 2 e consiste da resolução de Captchas com imagens de texto por parte de sistemas automatizados (19).

Figura 2: Captcha moderno que tenta dificultar o reconhecimento da imagem por sistemas automatizados através da distorção do texto e adição de uma linha que dificulta a segmentação dos caracteres

1.1 Classificação e Segmentação

A identificação de objetos em imagens consiste de duas principais subtarefas: a segmentação do objeto e a sua classificação. A segmentação, como o nome sugere, consiste em localizar o objeto na imagem, enquanto a classificação consiste em diferenciar a classe do objeto, de acordo com as classes reconhecidas pelo programa. No caso do exemplo da figura 2, podemos imaginar cada objeto como sendo uma letra diferente e cada classe como a letra do alfabeto que ela representa. Isso significa que as duas letras *i* do Captcha seriam dois objetos diferentes, porém, de mesma classe.

As Redes Neurais Convolutivas (CNNs) são utilizadas em ambas tarefas, com algumas diferenças estruturais em cada uma delas. No caso da classificação, o programa deve dizer apenas a classe do objeto, enquanto na segmentação, o programa deve dizer se para cada pixel na imagem, aquele pixel pertence ao objeto ou não. Na maioria dos casos, a segmentação é feita em conjunto com a classificação, indicando não apenas se o pixel pertence ao objeto, mas também à qual classe de objeto ele pertence. Por conta disso, é trivial perceber que o problema de segmentação é muito mais complexo que o de classificação sozinho. Em uma imagem com diversos objetos é necessário que ambas tarefas sejam realizadas para cada objeto. Porém, em uma imagem com um único objeto a segmentação pode ser excluída já que não existem outros objetos com os quais o programa pode se confundir.

1.2 Motivação e Objetivo

Este projeto visa utilizar as CNNs para criar um classificador que realize a tarefa de classificação de objetos em imagens de contexto real. A definição de contexto real é subjetiva, porém, imagens em contexto real são denominadas desta forma por apresentarem diversos problemas de processamento de imagens ao mesmo tempo. Estes problemas são exemplificados na figura 3 que consiste de uma cena de contexto real com vários objetos aparecendo em uma única imagem, onde estes podem sofrer oclusões, diferentes ângulos de perspectiva, luminosidade e muitos outros problemas em imagens desse tipo de contexto.

Sendo assim, os algoritmos que antes alcançavam resultados excelentes para bancos de imagens mais simples, como o CIFAR-10 (8) ou o PASCAL VOC2012 (4), agora precisam ser novamente aprimorados para obter a mesma perfomance em bancos de imagens em contexto real como o MS COCO. A figura 4 exemplifica as diferenças de complexidade entre as imagens dos 3 bancos. O banco CIFAR-10 possui imagens padronizadas de tamanho 32x32 com apenas um único objeto. O banco PASCAL VOC possui imagens em formas de fotografias com representações dos objetos em alta qualidade. E por sua vez, o banco MS COCO possui objetos em diferentes ângulos, perspectivas, e muitos outros problemas em imagens de tamanhos variados. Um banco de imagens pode ser chamado de *dataset*, apesar dessa nomeclatura também se estender para qualquer banco de dados.

1.3 Dataset

As imagens a serem utilizadas nesse projeto pertencem ao banco de dados do Challenge MS COCO 2015 (12) que representam cenas de 91 categorias de objetos. Apesar de existirem 91 categorias no banco do COCO, apenas 80 delas são classes diferentes enquanto as restantes são supercategorias utilizadas para agrupar os objetos. As supercategorias incluem: veículos, animais, objetos de higiene, objetos de esporte, entre outras.

O banco do MS COCO fornece, além da classe do objeto, anotações para ambos o *bounding box* (retângulo englobante) e a máscara de segmentação (delimitação manual do contorno) do objeto. A classificação do objeto pode ser feita tanto com o *bounding box* quanto com a máscara de segmentação. A figura 5 demonstra a diferença entre as anotações das máscaras de segmentação e as *bounding boxes*. As anotações não incluem dados de quais objetos estão se sobrepondo ou qualquer outra informação que facilite a classificação, os dados consistem apenas nos conjuntos de pontos que delimitam as *bounding boxes* ou

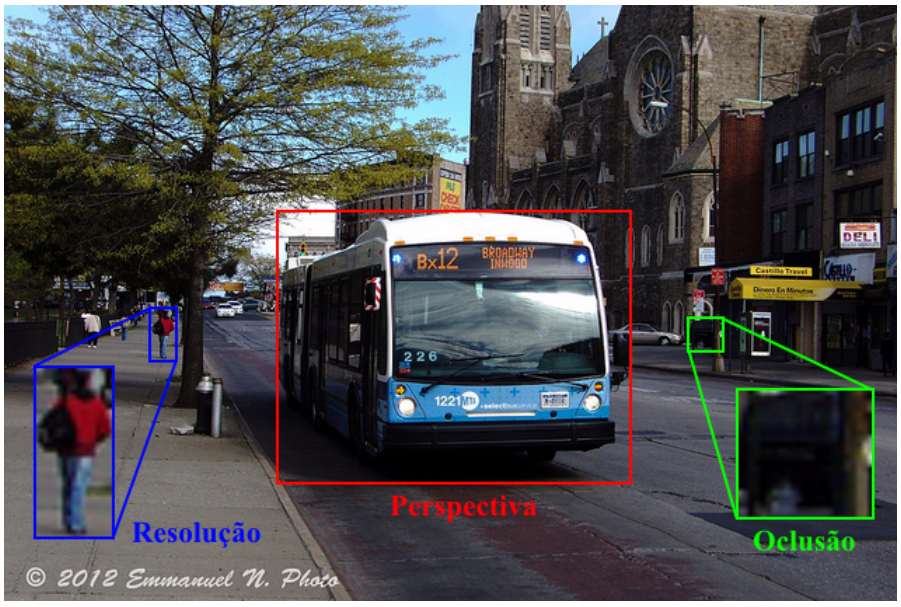

Figura 3: Exemplo de imagem de contexto real do banco MS COCO. Notam-se várias dificuldades para o reconhecimento dos objetos, como por exemplo, a baixa resolução das sub-imagens de pessoas por conta da distância para a câmera, o ângulo de perspectiva do ônibus, e a oclusão dos carros pelos prédios.

máscaras de segmentação e a classe a qual o objeto pertence. Demais anotações do COCO não são utilizadas nesse projeto.

O objetivo do projeto não inclui a segmentação dos objetos na imagem. Isso significa que é necessário que o classificador tenha as anotações dos objetos tanto durante o treino quanto durante a predição das classes.

É trivial notar que as máscaras de segmentação disponibilizam informações mais precisas sobre a delimitação dos objetos. Em uma imagem com muitos objetos, a *bounding box* sofrerá bastante superposição das anotações enquanto as máscaras de segmentação, apesar de também poderem sofrer superposição, são delimitadas com mais rigor e portanto funcionam melhor para esse tipo de imagem. A figura 6 demonstra a delimitação por máscara de segmentação como polígonos de cores diferentes para cada objeto.

1.4 Estado da Arte

O Challenge do MS COCO é um desafio criado para o *dataset* do COCO que foi realizado inicialmente em 2014. É possível estabelecer um estado da arte para este projeto baseando-se nos vencedores do Challenge de 2015. No entanto, vale a pena ressaltar que o Challenge consiste nas tarefas de segmentação e classificação, enquanto que este projeto tenta resolver apenas a tarefa de classificação a partir das anotações dos dados. A tabela 1 é o resultado do

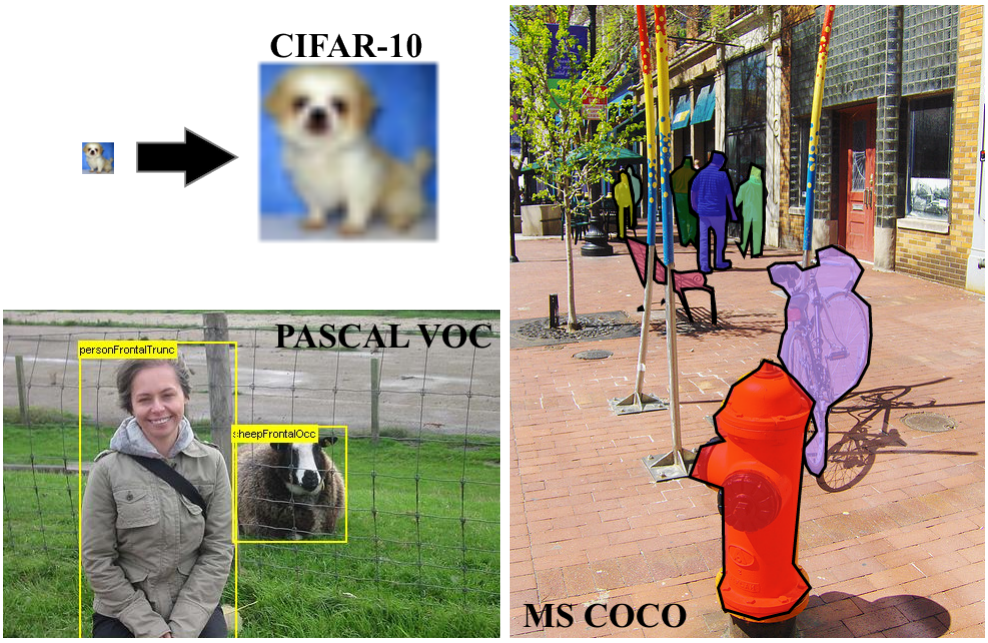

Figura 4: As diferenças entre imagens de contexto real do MS COCO e outros bancos de processamento de imagens.

Challenge 2015, com os 10 melhores grupos e seus respectivos resultados. A coluna AP indica a precisão média entre todas métricas utilizadas no Challenge para ambas tarefas de segmentação e classificação. O grupo MSRA (7) foi o que obteve os melhores resultados entre todos competidores. Para mais detalhes sobre as métricas, consulte (13).

Para obter esse resultado, esse grupo utilizou camadas residuais em conjunto com as convoluções. Assim, essa nova rede, denominada ResNet, não mapeia diretamente os inputs das camadas anteriores para camadas de convolução, mas sim para as camadas residuais que funcionam como funções não-lineares das camadas convolutivas.

Segundo experimentos realizados por eles, essas novas camadas residuais ajudam a melhorar a taxa de degradação das camadas mais profundas da rede. Eles realizam uma comparação de mesmo número de camadas (34) de uma arquitetura de CNN convencional com a ResNet proposta e conseguem obter uma precisão de resultados melhor utilizando a mesma quantidade de atributos.

Como o Challenge consiste principalmente da segmentação dos objetos, não há resultados disponíveis para apenas a tarefa de classificação dos objetos. A tabela 2 mostra os resultados da acurácia média de segmentação do grupo MSRA para 10 classes diferentes, escolhidas arbitrariamente, de modo a serem compararados com os resultados obtidos nesse projeto.

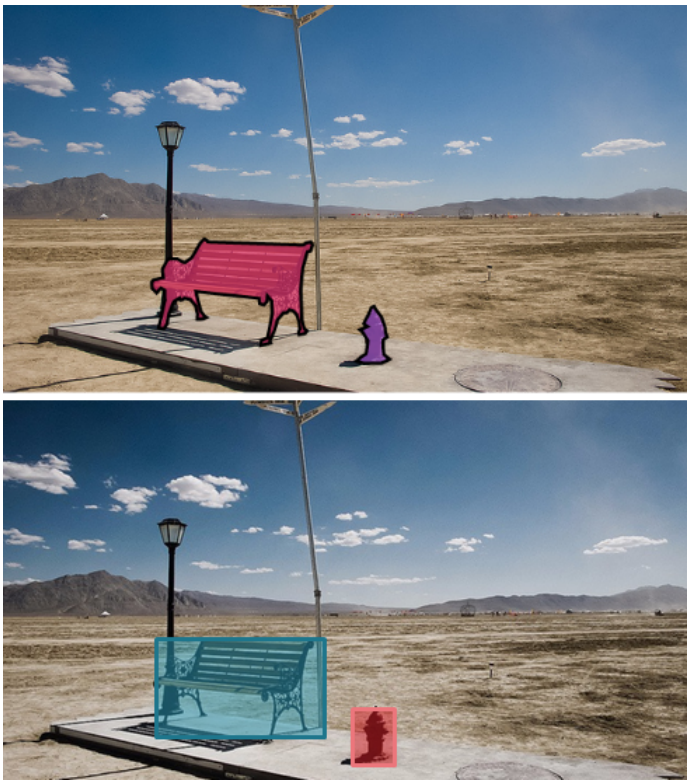

Figura 5: Em cima, uma imagem com suas anotações de máscaras de segmentação. Abaixo, a mesma imagem com suas anotações de *bounding boxes*.

1.5 Estrutura da Dissertação

Os capítulos a seguir estão organizados da seguinte forma: o capítulo 2 apresenta todos os conceitos e técnicas utilizadas para a implementação do classificador, o capítulo 3 explica como foi feita essa implementação e quais resultados foram obtidos a partir dela, e por fim, o capítulo 4 comenta as conclusões obtidas e possíveis trabalhos futuros.

Figura 6: Diferença entre uma imagem com e sem seus rótulos de máscara de segmentação

Grupo	**AP**
MSRA	0.373
FAIRCNN	0.335
ION	0.310
CMU_A2	0.257
UofA	0.255
Decode	0.224
wall	0.205
SinicaChen	0.190
UCSD	0.188
1026	0.179

Tabela 1: Lista de competidores do MS COCO Challenge 2015.

Classe	**AP**
giraffe	0.642
cat	0.639
dog	0.616
horse	0.574
airplane	0.560
sheep	0.439
truck	0.373
bird	0.340
car	0.328
bicycle	0.308

Tabela 2: Resultados do grupo MSRA no Challenge 2015 para a segmentação da *bounding box* em 10 classes diferentes escolhidas de forma arbitrária.

2
Conceitos

Neste capítulo descrevemos os conceitos teóricos e as técnicas utilizadas para a realização deste trabalho.

2.1
Redes Neurais

Uma Rede Neural, ou NN, é um modelo matemático inspirado pelo sistema nervoso central construído para resolver problemas de forma generalizada a partir da detecção de padrões. Uma NN possui diversos neurônios, originalmente conhecidos como unidades de McCullock - Pitts (14). Cada um deles, exemplificados na figura 7, realiza o somatório de suas entradas multiplicadas por seus respectivos pesos, conforme indica a equação 2-1, computando uma função de ativação em cima do resultado desse somatório para dizer se o neurônio foi ativado ou não.

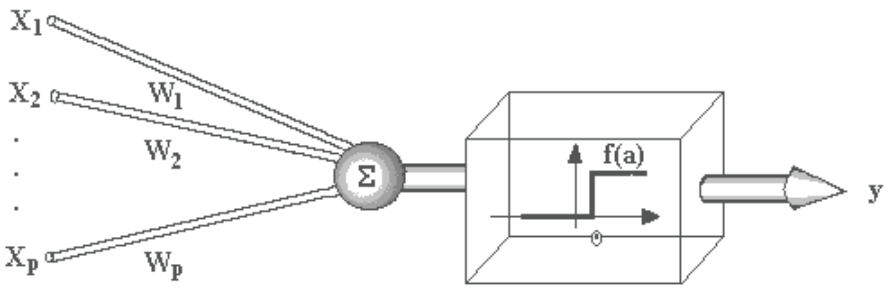

Figura 7: Esquema da unidade McCullock - Pitts, modernamente conhecido como um neurônio artificial.

$$a = X_1 \times W_1 + X_2 \times W_2 + ... + X_p \times W_p \tag{2-1}$$

Os neurônios podem ser aglomerados em grupos que denominamos de camadas. Cada camada normalmente possui visibilidade apenas para um grupo específico de entradas, mas não necessariamente utiliza todas as entradas visíveis. As camadas que lidam diretamente com o problema a ser solucionado, recebendo seus dados de entrada e definindo uma saída, são chamadas de

camadas externas, enquanto as demais camadas são chamadas de camadas internas.

Dizemos que um certo modelo de NN consegue aprender quando ele atinge uma solução generalizada para uma determinada classe de problemas. O aprendizado pode ser feito de diversas formas, mas a mais comum entre elas é o Backpropagation (20).

2.2 Feed-Forward

Uma NN é dita Feed-Forward (14) quando as conexões dos neurônios não formam ciclos e seguem sempre uma mesma direção. Sendo assim, a primeira camada de neurônios possui visibilidade apenas dos dados de entrada, a segunda camada possui visibilidade apenas das saídas da primeira camada, a terceira camada possui visibilidade apenas das saídas da segunda camada, e assim em diante até a última camada que fornece a(s) saída(s) da rede. A figura 8 exemplifica uma Feed-Forward NN de apenas 3 camadas, que também pode ser chamada de Perceptron Multicamadas, ou MLP.

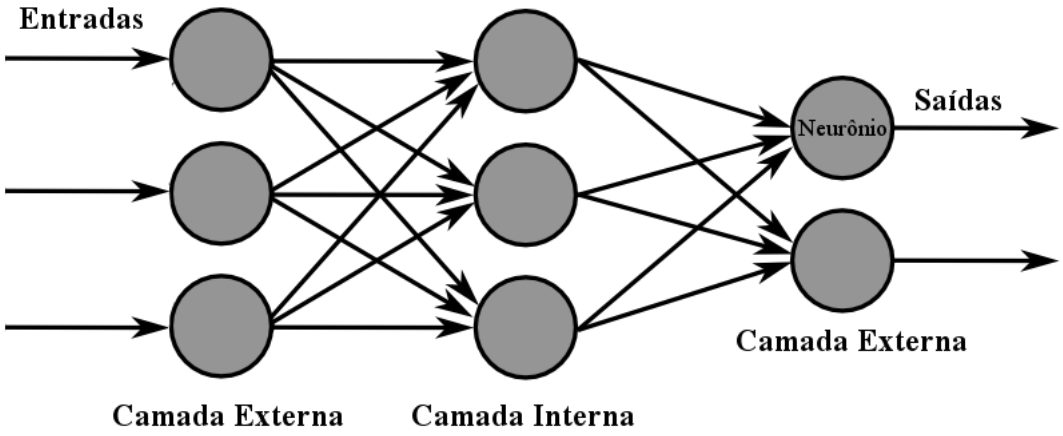

Figura 8: Uma Rede Neural Feed-Foward com 3 camadas totalmente conectadas.

2.3 Função de Ativação

Como já mencionado anteriormente, cada neurônio possui uma função de ativação (21) que avalia suas entradas e gera uma saída. O objetivo dessa função é introduzir uma não-linearidade na rede de forma que a saída final não possa ser reproduzida como uma recombinação linear dos dados de entrada. Para fins de simplificação do modelo, é comum que todos os neurônios de uma

determinada camada possuam a mesma função de ativação. Existem diversas funções de ativação diferentes, entretanto, as mais comuns em CNNs são as descritas abaixo para uma entrada x:

$$f(x) = \begin{cases} 0 & para \quad x < 0 \\ 1 & para \quad x \geq 0 \end{cases} \tag{2-2}$$

A função de Heaviside 2-2 (21), também conhecida como função degrau, é uma das funções mais simples e indica se um neurônio está ativado ou não. Durante o treinamento, os ajustes feitos nos pesos das entradas da camada que utiliza essa função terão efeitos não tão eficientes e limitados por conta da descontinuidade acentuada do degrau.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2-3}$$

A função Sigmóide 2-3 (21) é mais suave por não apresentar a mesma descontinuidade que o degrau e gera uma saída mais precisa no intervalo de 0 até 1 que pode ser interpretada como a porcentagem de ativação, ou probabilidade de ativação do neurônio.

$$f(x) = max(0, x) \tag{2-4}$$

A função Retificadora 2-4 (15) é a função mais popular em CNNs por dois principais motivos: sua ativação esparsa e a redução de problemas com o gradiente durante o aprendizado (6). Quando a Retificadora é utilizada em uma unidade ou camada, denominamos essa unidade ou camada de Unidade Linear Retificada, ou ReLU.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2-5}$$

A função Softmax 2-5 (21) é utilizada para converter valores em um vetor z de dimensão K em outro vetor $\sigma(z)$ de dimensão K de valores reais entre 0 e 1. Essa função de ativação é comumente usada na última camada da rede de forma a gerar o vetor que indica qual classe o objeto pertence, ou seja, a classe de maior probabilidade.

2.4 Backpropagation

O Backpropagation (20) é um método de aprendizado supervisionado utilizado no treinamento de Feed-Forward NNs. O treinamento é chamado de supervisionado pois o modelo possui acesso às saídas corretas, também conhecidas como rótulos, para cada entrada durante a fase de treino. Treinar uma rede

de forma supervisionada significa ajustar os pesos que multiplicam as entradas de cada neurônio até que a saída dessa rede seja a mais próxima possível de seu rótulo. Além de precisar de rótulos para realizar o treino, o Backpropagation também necessita que as funções de ativação sejam diferenciáveis.

A comparação entre a saída predita de um modelo e seu rótulo é feita através do uso de uma função de perda. A função Mean Squared Error 2-6, ou MSE, exemplifica uma função de perda bastante comum que é utilizada nesse trabalho. Essa função calcula a média entre os quadrados das diferenças das saídas reais, ou rótulos, Y^{real} e as saídas preditas $Y^{predita}$, com uma quantidade de saídas n.

$$MSE = \frac{1}{n} \times \sum_{i=1}^{n} (Y_i^{real} - Y_i^{predita})^2 \tag{2-6}$$

O objetivo do Backpropagation é minimizar essa função de perda, calculando a derivada parcial dos erros de cada neurônio em relação às suas entradas $\frac{\partial E}{\partial w}$ e tentando achar um mínimo local a partir do gradiente dessa derivada.

2.5 Hiperparâmetro

Em aprendizado de máquina, denominamos de hiperparâmetro (21) uma variável de valor estabelecido de forma manual pelo modelador da rede que não se altera durante a etapa de treinamento. As demais variáveis, que atualizam seus valores ao longo do treino, são denominadas parâmetros da rede. Os hiperparâmetros são calibrados de forma arbitrária seguindo hipóteses, intuições e conhecimentos prévios do modelador.

2.6 Overfitting

Dizemos que um modelo está sofrendo overfitting (21) quando ele perde sua capacidade de generalização e predição dos dados. Em termos informais, dizemos que um modelo está overfitting quando ele deixa de "aprender"sobre os dados de treino e passa a "memorizá-los". Isso significa que o modelo possui alta precisão durante a fase de treinamento mas perde precisão durante a predição dos dados de teste.

2.7 Instância

Uma instância (21) é um dado de entrada para a rede neural. Para esse projeto, uma entrada consiste em todo conjunto de *pixels* contido em uma *bounding box*, ou seja, um objeto e seu *background*.

2.8 Métricas

Métricas (21) são ferramentas estatísticas usadas para comparar e avaliar classificadores. Nesse projeto são utilizadas apenas as métricas de *precision* (precisão), *recall* e F_1.

As nomenclaturas VP, FP e FN indicam os verdadeiros positivos, falsos positivos e falsos negativos, respectivamente. Para explicar cada conceito imaginemos a seguinte situação: uma imagem possui diversas classes de objetos, incluindo gatos. Os verdadeiros positivos são os gatos preditos corretamente como gatos, os falsos positivos são os gatos preditos erroneamente como qualquer outra classe e os falsos negativos os preditos como gatos que não são gatos.

A *precision* 2-7 indica a fração de prediçoes corretas dentro do conjunto de prediçoes. O *recall* 2-8 indica a fração de predições corretas dentro do conjunto de todos objetos da classe. E por sua vez, o F_1 2-9 é a métrica que relaciona as duas outras métricas descritas.

$$precision = \frac{VP}{VP + FP} \tag{2-7}$$

$$recall = \frac{VP}{VP + FN} \tag{2-8}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2-9}$$

2.9 Redes Neurais Convolutivas

A Rede Neural Convolutiva (10) (3), ou CNN, é um modelo específico de NN feed-forward que utiliza o recurso das convoluções.

Cada neurônio realiza um produto escalar em suas entradas e opcionalmente também realiza uma função não-linear antes de gerar uma saída. Mais precisamente, as CNNs são modelos de Deep Neural Networks (16) (5), ou DNNs, que funcionam melhor para o processamento de imagens pois as convoluções realizadas por elas representam as informações que um *pixel* contém relacionado com seus vizinhos.

As DNNs convencionais possuem todas as conexões possíveis entre cada *pixel* (uma entrada) e um neurônio. Isso significa que essa rede é configurada de forma totalmente conectada, conforme mostra a figura 9. Essa configuração, além de custosa, não prioriza *pixels* de posições vizinhas e faz com que diversos atributos extraídos pela rede sejam de pouca importância.

No caso das CNNs, as camadas são arrumadas de forma que cada uma delas possua neurônios agrupados em 3 dimensões teóricas: altura, largura e profundidade, conforme explica a figura 10. A entrada da CNN é um volume que possui dimensões iguais à altura e largura da imagem, caso esta não seja redimensionada, e profundidade igual à quantidade de canais de cores da imagem. Na tarefa de classificação, a saída da CNN é um vetor binário que identifica a classe do objeto na imagem e possui uma profundidade igual ao número de classes do problema, com as demais dimensões iguais a 1.

A extração dos atributos, agora feita pelas camadas convolutivas, leva em conta a disposição espacial dos valores nos volumes de entrada. Essa configuração faz com que a extração de atributos leve em conta apenas *pixels* em posições vizinhas. Isso significa que as CNNs conseguem extrair de forma inteligente atributos de uma determinada imagem, diferentemente das NNs convencionais. Portanto, as CNNs extraem menos atributos, e ao mesmo tempo, atributos mais importantes para as tarefas de visão computacional.

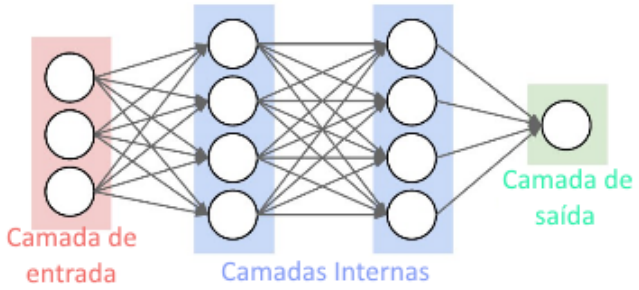

Figura 9: Uma DNN convencional que possui suas camadas representadas por neurônios em 2 dimensões.

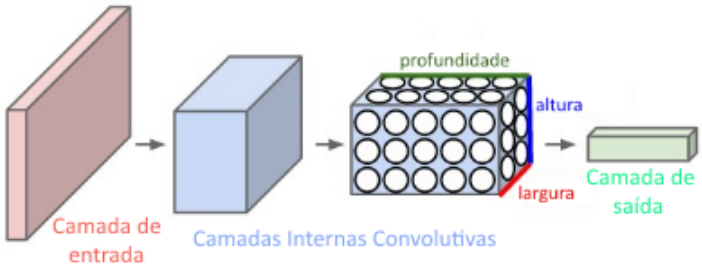

Figura 10: Uma CNN que possui suas camadas representadas por neurônios em 3 dimensões.

2.9.1 Camada Convolutiva

A camada convolutiva (10), também conhecida como camada de convolução, é responsável por extrair os atributos que definem a classe do objeto. Cada camada convolutiva possui uma quantidade K de filtros de duas dimensões, onde cada dimensão tem tamanho F. Em outras palavras, podemos dizer que cada filtro é uma matriz e existem K matrizes de dimensão $F \times F$. A quantidade K de filtros é igual à profundidade da entrada dessa camada.

Cada filtro está associado a uma região, denominada campo receptivo, da entrada da camada. Esse campo receptivo possui as mesmas dimensões F de altura e largura que o filtro. Cada neurônio da camada convolutiva recebe o resultado do somatório dos produtos escalares de cada campo receptivo com seu filtro ao longo da profundidade da entrada. Esse resultado, denominado a, pode ser visualizado na equação 2-10, onde K é o número de filtros e X_i e W_i referem-se à entrada e ao filtro de posição i, respectivamente. A figura 11 exemplifica esse cálculo utilizando um filtro de tamanho F igual ao tamanho da entrada e profundidade $K = 2$.

$$a_i = \sum_{i=0}^{K} X_i \cdot W_i \tag{2-10}$$

Entretanto, é comum que o filtro seja de tamanho menor que a entrada. Nesse caso, é preciso iterar o campo receptivo por toda extensão bidimensional da altura e largura da entrada, em uma operação que é chamada de convolução. O número de posições que o campo receptivo se locomove a cada iteração é chamado de *stride* e é denominado pelo hiperparâmetro S. A figura 12 mostra um exemplo de convolução de um filtro de tamanho $F = 3$ e $K = 1$ em uma entrada de dimensões $5 \times 5 \times 1$ utilizando um *stride* $S = 1$.

2.9.2 Zero-Padding

O Zero-Padding (11) é definido como o hiperparâmetro P e ele define a quantidade de zeros que serão acrescentados nas bordas da entrada de forma a manipular suas dimensões de altura e largura. Um valor de $P = 2$ indica que dois valores de zero serão acrescentados em cada posição da borda da entrada, conforme demonstra a figura 13. Além de possibilitar a manipulação das dimensões da entrada, o Zero-Padding também conserva os valores das posições perto das bordas. É possível definir o tamanho da saída da camada convolutiva utilizando a fórmula $(W - F + 2P)/S + 1$ onde W é o tamanho da entrada, S o *stride*, F o tamanho do filtro, e P o Zero-Padding.

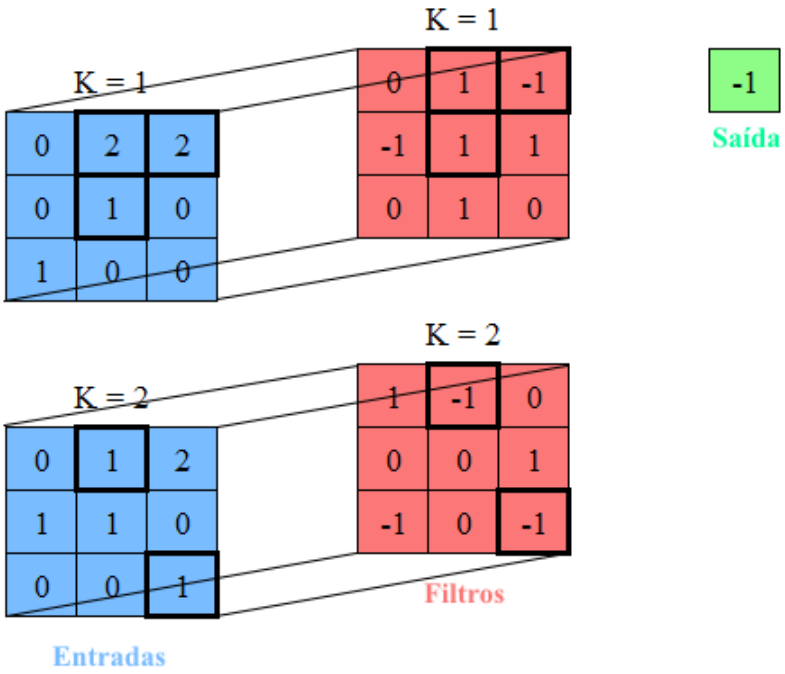

Figura 11: Um exemplo de cálculo do a de um neurônio. Para realizar o cálculo, basta multiplicar os valores das entradas com os valores nos filtros nas respectivas posições e realizar a soma de todos eles. Neste caso, $a = 2 \times 1 + 2 \times (-1) + 1 \times 1 + 1 \times (-1) + 1 \times (-1) = -1$.

2.9.3 Dropout

O Dropout (17) é uma técnica de regularização utilizada com o intuito de reduzir o overfitting da rede durante o treino. O dropout é aplicável em qualquer camada de uma rede e consiste em desativar os neurônios dessa camada com uma certa probabilidade p. Isso significa que para uma mesma quantidade de parâmetros em uma camada, a rede será treinada diversas vezes.

2.9.4 Camada ReLU

A camada ReLU normalmente segue após a camada convolutiva e utiliza a saída a da camada convolutiva aplicada na função de ativação ReLU para definir a probabilidade de ativação de cada neurônio.

2.9.5 Camada de Pooling

O objetivo da Camada de Pooling é reduzir as dimensões da entrada de modo a minimizar o número de parâmetros computados pela rede e controlar o overfitting. O tipo de pooling mais utilizado é o pooling de máximo, ou Max

Pooling, que seleciona o maior valor dentro de uma região da entrada. De forma similar à camada convolutiva, o pooling é aplicado a um campo receptivo de tamanho F e possui um *stride* S. A figura 14 exemplifica um Max Pooling com $F = 2$ e $S = 2$.

2.9.6 Camada Densa

A camada densa é uma camada comum totalmente conectada de uma DNN convencional. O parâmetro n define o número de neurônios nessa camada.

2.10 Redes Neurais Residuais

Também conhecidas como ResNets (7), essas redes se diferenciam das CNNs convencionais pois incorporam uma técnica para evitar a degradação de dados nas camadas mais profundas da rede e o desaparecimento do gradiente durante o Backpropagation. Uma ResNet possui camadas residuais que são compostas, cada uma, de duas camadas convolutivas e duas funções de ativação ReLU, conforme demonstra a figura 15. A camada residual difere das demais camadas convolutivas pois sua saída é a soma da saída da segunda camada convolutiva com a entrada da camada residual. Essa estrutura possibilita que camadas mais profundas recebam diretamente dados das camadas mais superficiais, sem a necessidade de passar por camadas convolutivas intermediárias.

2.11 Pré-Processamento

As técnicas de pré-processamento são operações realizadas nas imagens antes que estas sejam enviadas para a rede. Quando um pré-processamento é realizado com o objetivo de aumentar o número de exemplos ou facilitar a extração de atributos durante o treino, ele é chamado de Data Augmentation (9). Essas operações são realizadas para melhorar o desempenho da rede, evitando o overfitting, sem que haja a necessidade de criar novos dados de entrada com novas anotações.

2.11.1 Redimensionamento

O redimensionamento, ou Resize, consiste em modificar as dimensões de altura e largura da imagem para padronizar o tipo de entrada da rede.

É comum que as dimensões de altura e largura sejam redimensionadas para valores iguais e múltipos de 2 de modo a facilitar as operações de pooling.

2.11.2 Flip

A técnica do Flip Horizontal realiza o espelhamento horizontal na imagem de modo a aumentar a quantidade de exemplos durante o treino. É importante mencionar que o Flip Vertical raramente faz sentido para objetos que possuam assimetria vertical. O flip geralmente é aplicado com uma probabilidade p de ocorrência.

2.11.3 Shift

O shift é uma técnica de data augmentation que realiza a translação da imagem em uma determinada direção, podendo ela ser horizontal, vertical ou ambas. Esse shift ocorre com uma probabilidade p e translada o objeto em uma distância w proporcional à distância do objeto com a borda da imagem.

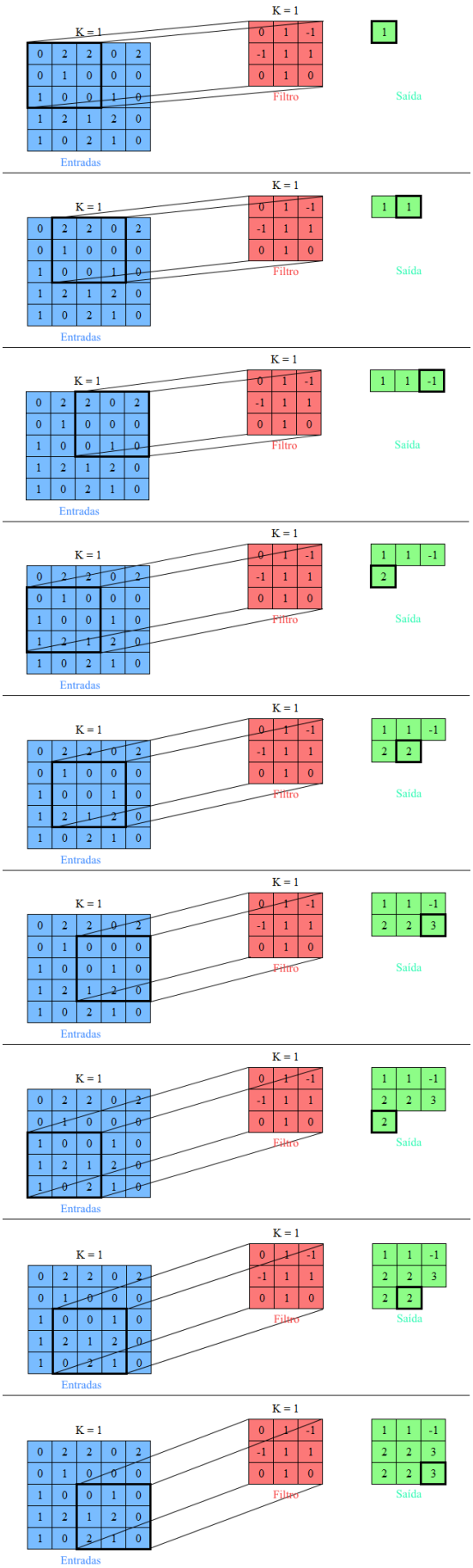

Figura 12: A convolução de um filtro de tamanho $F = 3$ em uma entrada de dimensão $5 \times 5 \times 1$ com $S = 1$. Cada linha indica uma iteração da convolução.

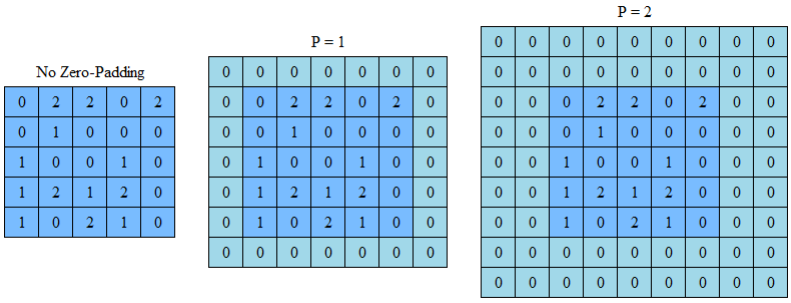

Figura 13: Um exemplo de entrada com diferentes valores de Zero-Padding.

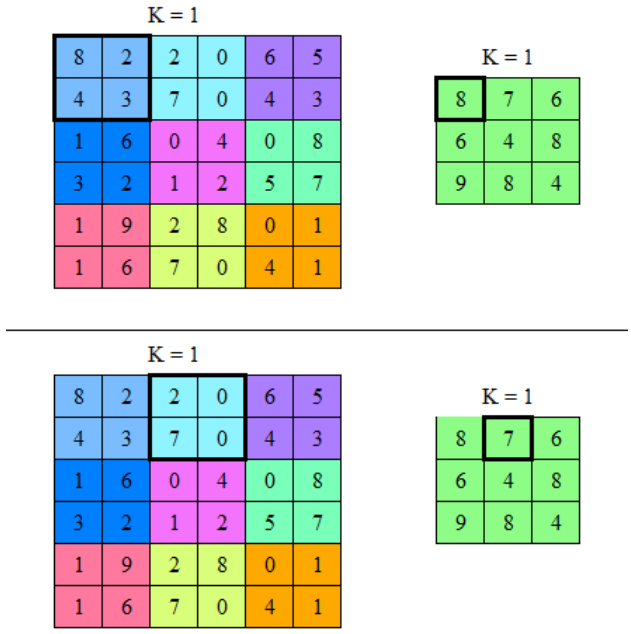

Figura 14: Um exemplo com duas iterações de um Max Pooling com $F = 2$ e $S = 2$. Cada cor diferente mostra um campo receptivo do pooling.

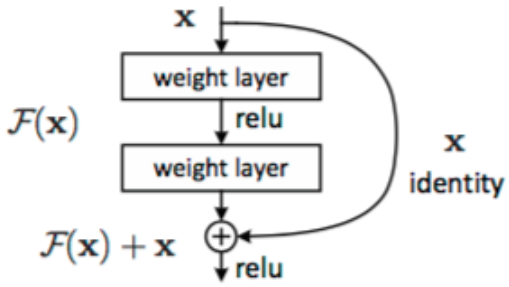

Figura 15: Uma camada residual é composta de duas camadas de convolução com uma função de ativação ReLU entre elas. A saída da camada residual é a função ReLU da sua própria entrada acrescentada da saída da segunda camada intermediária.

3
Metodologia e Resultados

A implementação foi realizada em Python com a ferramenta Theano (1), que gera código CUDA para o processamento rápido em GPU, essencial para aplicações como esta que demandam uma carga bastante elevada de processamento. Além disso, também foi utilizado o Keras (2), uma ferramenta que gera código Theano para a abstração da modelagem de aplicações de machine learning.

A tarefa de classificação foi realizada por uma ResNet e uma CNN, utilizando como base redes estabelecidas para outros *datasets* como o do CIFAR-10 (8) (9). Os conceitos descritos no capítulo 2 foram utilizados nos experimentos de forma a otimizar o resultado alcançado.

Como existem apenas anotações para os conjuntos de treino e validação no *dataset* do MS COCO, o conjunto de validação é separado e utilizado como conjunto de teste para o programa e o conjunto original de teste do *dataset* é descartado. Denominamos de *train split* quando o conjunto de treino do COCO é separado com 70% das imagens para treino e 30% para validação do programa. Denominamos de *test split* quando 100% do conjunto de treino do COCO é usado para treino e 100% do conjunto de validação do COCO é usado para teste no programa. Essa separação dos conjuntos é feita de forma a garantir que o conjunto de testes não seja utilizado durante a calibração dos hiperparâmetros do algoritmo, prática comum em aprendizado de máquina (21). É importante também mencionar que é realizado um *shuffle* (21) dentro de cada conjunto, modificando a ordem de entrada das instâncias para a rede de modo a garantir a qualidade do algoritmo.

Um filtro que obtém apenas imagens que possuem objetos de determinadas classes também foi realizado. Esse filtro foi construído de forma a reduzir o tempo de treino e validação durante a fase de ajuste de hiperparâmetros ou comparação de diferentes modelos de rede. Se necessário, o treinamento pode ser particionado em diversos *batches*, possibilitando que todo conjunto de treino seja utilizado mesmo que este não caiba por inteiro na memória do computador.

3.1 Arquitetura 1 - CNN

A primeira arquitetura consiste de uma CNN sem camadas residuais e de poucas camadas convolutivas. Mais especificamente, esse modelo possui 4 camadas de convolução, 2 camadas totalmente ligadas e 2 camadas de generalização Max Pooling. A maioria das camadas utiliza a função de ativação ReLU, exceto pela última camada, que utiliza a Softmax para classificar o output em uma única classe. Também foram utilizados dropouts de modo a evitar o overfitting da rede. A função de otimização utilizada é do gradiente adaptativo Adagrad (22). A figura 16 demonstra a CNN dessa arquitetura.

O objetivo dessa arquitetura é comparar diferentes técnicas de aumento de *dataset* e a complexidade de diferentes *datasets*. Essa arquitetura é simples de modo a garantir que a comparação seja feita de forma ágil.

3.2 Arquitetura 2 - ResNet

A arquitetura 2 foi obtida com base no modelo utilizado no estado da arte. Ela consiste de uma rede residual com 9 camadas de convolução, onde 8 delas são seguidas por uma camada residual, 3 camadas de pooling, sendo que uma delas utiliza o pooling médio ao invés do pooling máximo, e 1 camada totalmente ligada para gerar a classificação da instância. A técnica do dropout também é utilizada como na arquitetura 3.1, porém, de forma menos excessiva. A função de otimização utilizada também é o Adagrad. E por fim, a figura 17 demonstra essa arquitetura explicitando a ordem das camadas e seus respectivos parâmetros. Cada camada residual é composta de duas camadas convolutivas idênticas e duas funções de ativação, conforme descrito no capítulo 2.10.

O objetivo dessa arquitetura é maximizar o resultado das métricas de avaliação do classificador. Essa arquitetura possui um tempo de execução elevado por conta de sua complexidade. Sendo assim, ela é utilizada principalmente para exibir os resultados finais do programa.

3.3 Pré-Processamento e Aumento do Dataset

Neste projeto foram utilizadas ambas as anotações dos *bounding boxes* e as máscaras de segmentação dos objetos. Cada instância é recortada no tamanho de sua *bounding box* e redimensionada para o tamanho padrão de entrada de 32x32 *pixels* que é fornecido para a rede. Se habilitada, a máscara de segmentação é utilizada para separar a instância de seu plano de fundo.

Essa separação é realizada de forma a preencher o plano de fundo do *bounding box* com *pixels* pretos, destacando o objeto e facilitando o aprendizado. Em alguns modelos, é utilizado um *padding* adicional de pré-processamento de instâncias menores que 32x32, centralizando a instância em sua *bounding box*, preenchendo o restante da sub imagem com *pixels* pretos, e evitando distorções no redimensionamento para o tamanho maior padronizado. A figura 18 demonstra de forma visual o pré-processamento realizado em cada instância antes de sua alimentação para a rede.

Abaixo estão listados os experimentos realizados que verificam hipóteses e fornecem dados sobre os resultados obtidos no projeto. Todos os experimentos, exceto do experimento 1, utilizam as seguintes técnicas de pré-processamento:

- O flip horizontal, aplicado com uma probabilidade $p = 0.5$.
- O Zero-Padding antes das camadas convolutivas, com tamanho relativo à saída da camada anterior.
- O shift horizontal com probabilidade $p = 0.5$ e uma distância percentual $w = 0.2$.

3.4 Testes Automatizados

Testes unitários foram realizados em cada módulo e cada função utilizando o Python Unit Framework em conjunto com o módulo de testes fornecido pelo pacote Numpy. Os testes foram realizados de forma independente, entretanto, algumas funções necessitaram da chamada de outras funções para a formatação dos dados de input.

Não foi possível realizar o teste de corretude no modelo de rede neural convolutiva gerada pela ferramenta Keras pois ela não fornece módulos de teste para os modelos gerados. O Keras é relativamente novo, e portanto, ainda encontra-se em estado de desenvolvimento.

Assertivas foram implementadas em cada função de modo a garantir os tipos e a formatação dos dados de entrada. Entretanto, em alguns casos não é possível garantir a corretude dos dados de entrada quando a aplicação trabalha com volumes muito grandes de dados. Ao carregar o conjunto de treino ou validação na memória, as funções não podem ficar constantemente reavaliando se cada dado individual possui o tipo e formatação correta. Sendo assim, as assertivas de corretude foram implementadas na medida do possível de forma a não penalizar o desempenho do sistema. Nesses casos, a garantia de corretude é feita apenas nas funções que carregam os dados na memória.

3.5 Experimento 1 - Comparação: CIFAR-10 vs COCO

3.5.1 Descrição

O primeiro experimento a ser realizado consiste em uma comparação entre a complexidade do *dataset* do COCO, que é formado por imagens em contexto real, e o CIFAR-10, que utiliza imagens simples e já redimensionadas. A comparação é feita através de uma CNN simples, utilizando a arquitetura de rede definida em 3.1, e o *train split*. Essa rede não utiliza nenhuma etapa de pré-processamento além do redimensionamento das instâncias para o tamanho padronizado. Como cada *dataset* possui uma quantidade diferente de categorias, são escolhidas apenas 10 categorias do banco do COCO de modo a simular as categorias do CIFAR-10. O objetivo desse experimento é comparar a complexidade entre os dois *datasets*.

3.5.2 Resultados

Utilizando 40 épocas, obtemos o resultado de precisão média no *dataset* do CIFAR-10 em 82%. Em contraste, o mesmo modelo, utilizando a mesma quantidade de épocas, obtém uma precisão média de 59% no banco de dados do COCO, conforme indica a tabela 3.

	precision	**recall**	F_1	**support**
car	0.70	0.88	0.78	23236
sheep	0.63	0.62	0.63	4516
giraffe	0.56	0.70	0.62	1909
airplane	0.63	0.50	0.56	2122
bicycle	0.55	0.51	0.53	3986
bird	0.56	0.45	0.50	4874
horse	0.51	0.43	0.47	2756
truck	0.47	0.22	0.30	6437
cat	0.33	0.37	0.35	1904
dog	0.28	0.17	0.21	2559
avg / total	0.59	0.62	0.59	54299

Tabela 3: Resultados do experimento 1 no *dataset* COCO em diferentes métricas. Resultados em ordem decrescente de F_1.

O classificador consegue atingir um resultado melhor (59% de precisão média) do que um baseline que seria de classificar sempre a classe com maior ocorrência (41% de precisão média). Isso significa que o modelo consegue

aprender mesmo em um *dataset* de complexidade elevada como o do MS COCO.

3.6 Experimento 2 - Aumento de dataset e padding de instâncias pequenas

3.6.1 Descrição

O experimento 2 é uma extensão do primeiro experimento que visa comparar os resultados no *dataset* COCO quando técnicas de pré-processamento são acrescentadas. Utilizando o *train split* com o mesmo modelo que o primeiro experimento, agora são aplicadas três técnicas de aumento do *dataset*: o *padding* de instâncias pequenas, o flip-horizontal, e o shift horizontal nas instâncias do conjunto de treino. Além disso, também é realizado o Zero-Padding dos inputs de cada camada convolutiva, seguindo a estratégia adotada em (11). O objetivo desse experimento é comprovar que essas técnicas são eficientes e de fato melhoram o desempenho do classificador.

3.6.2 Resultados

	precision	**recall**	$\boldsymbol{F_1}$	**support**
car	0.73	0.93	0.82	14323
giraffe	0.73	0.85	0.79	1194
sheep	0.77	0.70	0.73	2608
bicycicle	0.73	0.68	0.70	2459
airplane	0.84	0.52	0.64	1601
horse	0.68	0.54	0.60	1781
bird	0.62	0.49	0.54	2709
cat	0.48	0.57	0.52	1207
truck	0.68	0.37	0.47	4129
dog	0.43	0.32	0.37	1502
avg / total	0.70	0.70	0.69	33513

Tabela 4: Resultados do experimento 2 no *dataset* COCO em diferentes métricas. Resultados em ordem decrescente de F_1.

A partir dos resultados obtidos na tabela 4, pode-se concluir que as técnicas de pré-processamento e o Zero-Padding nas convoluções elevaram o valor médio da precisão de 59% para 70% e o *recall* médio de 62% para 70%.

3.7 Experimento 3 - Comparação: CNN vs ResNet

3.7.1 Descrição

Esse experimento avalia a diferença de resultados ao utilizar a arquitetura 2 de ResNet ao invés da arquitetura 1 de CNN comum. O experimento foi realizado com o *train split* em apenas 10 classes de objetos de forma a ser concluído com rapidez em ambas arquiteturas. As classes escolhidas são as mesmas dos experimentos 1 e 2. A comparação foi realizada com as mesmas técnicas de pré-processamento descritas no experimento 2, em 40 épocas. Os resultados da ResNet obtidos na tabela 5 podem ser comparados com os resultados da CNN da tabela 4, entretanto, note que a ordem das linhas está diferente pois estas estão organizadas por F_1 em cada tabela. O objetivo desse experimento é comprovar que a arquitetura 3.2 de rede residual de fato obtém melhor desempenho que a arquitetura 3.1.

3.7.2 Resultados

	precision	**recall**	$\boldsymbol{F_1}$	**support**
car	0.82	0.89	0.85	14323
giraffe	0.82	0.84	0.83	1194
airplane	0.83	0.72	0.77	1601
bicycle	0.74	0.78	0.76	2459
sheep	0.74	0.76	0.75	2608
horse	0.64	0.66	0.65	1781
bird	0.61	0.62	0.61	2709
truck	0.71	0.52	0.60	4129
cat	0.53	0.51	0.52	1207
dog	0.44	0.38	0.41	1502
avg / total	0.74	0.75	0.74	33513

Tabela 5: Resultados do experimento 3 no *dataset* COCO em diferentes métricas. Resultados em ordem decrescente de F_1.

modelo	**precision**	**recall**	$\boldsymbol{F_1}$
ResNet	0.74	0.75	0.74
CNN	0.70	0.70	0.69

Tabela 6: Comparação entre a arquitetura 1 e a arquitetura 2, utilizando a média das métricas entre todas 10 classes.

Com a comparação dos resultados descritos na tabela 6, é possível afirmar que a rede residual da arquitetura 2 é melhor que a CNN da arquitetura 1.

3.8 Experimento 4 - Comparação: Bounding Box vs Máscara de Segmentação

3.8.1 Descrição

Esse experimento avalia as diferenças entre a utilização apenas da *bounding box* em relação à utilização da *bounding box* em conjunto com a máscara de segmentação. Seguindo o mesmo padrão dos experimentos anteriores, esse experimento também foi realizado com o *train split*. O objetivo desse experimento é avaliar se de fato o uso das máscaras de segmentação afetam o desempenho do classificador.

3.8.2 Resultados

Os gráficos 19 e 20 mostram os resultados da precisão e do *recall*, respectivamente. Como podemos observar, a máscara de segmentação é bem impactante na performance do algoritmo. Entretanto, vale a pena ressaltar que o uso da máscara de segmentação aumenta consideravelmente o tempo de execução, conforme mostra o gráfico 21. Apesar do tempo de execução elevado, optamos por utilizar a máscara de segmentação para a execução final do experimento 5.

3.9 Experimento 5 - Resultados Finais

3.9.1 Descrição

O experimento 5 é o mais importante para esse projeto pois ele utiliza todas as 80 classes de objetos existentes no banco do MS COCO. Os resultados foram obtidos após cerca de 11 dias de execução em uma GPU Tesla K40, incorporando a arquitetura 2 e todas técnicas de pré-processamento descritas até então. Por conta da grande quantidade de memória necessária para carregar o *dataset* por completo, utilizamos uma divisão do conjunto de treino em 10 partições diferentes, limitando os recursos utilizados em aproximadamente 32 GB de memória RAM. É importante ressaltar que esse experimento é o único que adota o *test split*, utilizando o conjunto de validação do COCO como conjunto de testes para o experimento.

3.9.2 Resultados

Os resultados finais do projeto se encontram na tabela 7 com todas as métricas para cada classe do banco.

	precision	**recall**	F_1	**support**
person	0.85	0.94	0.89	328561
car	0.68	0.80	0.74	58331
motorcycle	0.69	0.71	0.70	47746
parking meter	0.70	0.66	0.68	15494
stop sign	0.67	0.67	0.67	16260
airplane	0.63	0.68	0.65	45600
bus	0.61	0.68	0.65	44571
traffic light	0.66	0.66	0.65	17578
cell phone	0.67	0.61	0.64	2864
train	0.63	0.62	0.63	30539
oven	0.65	0.61	0.63	2474
truck	0.61	0.63	0.62	29480
keyboard	0.65	0.60	0.62	3051
book	0.72	0.54	0.61	1859
fire hydrant	0.60	0.61	0.60	16941
bench	0.62	0.58	0.60	14874
bicycle	0.55	0.63	0.59	71511
sheep	0.61	0.58	0.59	11705
skateboard	0.61	0.58	0.59	8156
cow	0.62	0.55	0.58	11436
elephant	0.64	0.54	0.58	11121
bear	0.60	0.58	0.58	10876
mouse	0.62	0.55	0.58	3317
zebra	0.59	0.57	0.57	10603
remote	0.62	0.55	0.57	3180
spoon	0.60	0.54	0.56	6828
frisbee	0.58	0.53	0.55	9264
fork	0.57	0.54	0.55	7203
carrot	0.60	0.52	0.55	5281
baseball bat	0.58	0.50	0.54	8459
cup	0.57	0.52	0.54	7415
microwave	0.57	0.51	0.54	2570
tie	0.56	0.52	0.53	9614

sports ball	0.55	0.51	0.52	8747
baseball glove	0.55	0.50	0.52	8292
hot dog	0.56	0.49	0.52	5171
surfboard	0.54	0.48	0.51	7937
knife	0.52	0.50	0.51	7008
orange	0.55	0.49	0.51	5514
giraffe	0.55	0.46	0.50	10387
tenns racket	0.52	0.48	0.50	7803
donut	0.59	0.43	0.50	4891
skis	0.50	0.49	0.49	9136
snowboard	0.54	0.46	0.49	8918
bowl	0.51	0.49	0.49	6465
laptop	0.52	0.48	0.49	3504
clock	0.63	0.42	0.49	1769
kite	0.50	0.47	0.48	8572
bottle	0.50	0.46	0.48	7717
broccoli	0.53	0.44	0.48	5374
chair	0.52	0.46	0.48	4640
refrigerator	0.61	0.43	0.48	1966
bird	0.55	0.42	0.47	14194
umbrella	0.53	0.43	0.47	9993
toilet	0.53	0.44	0.47	3840
tv	0.51	0.44	0.47	3728
backpack	0.51	0.44	0.46	10127
apple	0.51	0.44	0.46	6024
pizza	0.53	0.42	0.46	5057
cake	0.50	0.43	0.46	4758
wine glass	0.49	0.43	0.45	7538
couch	0.53	0.41	0.45	4527
horse	0.52	0.39	0.44	12242
handbag	0.49	0.41	0.44	9813
sandwich	0.52	0.40	0.44	5763
bed	0.48	0.41	0.44	4308
boat	0.46	0.40	0.43	19887
banana	0.50	0.38	0.43	6322
toaster	0.51	0.39	0.43	2290
suitcase	0.48	0.38	0.42	9437
dining table	0.49	0.37	0.42	4123
cat	0.48	0.36	0.41	13639

sink	0.54	0.37	0.41	2159
potted plant	0.48	0.35	0.40	4422
dog	0.49	0.33	0.39	13188
vase	0.47	0.22	0.29	1589
scissors	0.44	0.21	0.28	1430
teddy bear	0.50	0.18	0.25	576
hair drier	0.44	0.18	0.24	524
toothbrush	0.20	0.01	0.02	235
avg / total	0.66	0.67	0.66	1180306

Tabela 7: Desempenho da arquitetura 2 no *dataset* completo do MS COCO. Resultados em ordem decrescente de F_1.

Utilizando apenas as 10 classes definidas na tabela 2 e usadas como validação dos outros experimentos, podemos realizar uma comparação do resultado final obtido com o estado da arte do grupo MSRA. O gráfico 22 fornece informações de que a segmentação é menos eficiente para objetos pequenos. O gráfico 23 fornece uma visão mais generalizada da comparação entre todas as 80 classes. Podemos observar que nesse caso, a precisão média do experimento 5 é de 0.66, enquanto a do grupo MSRA é de 0.37. No entanto, devemos nos lembrar que essas comparações não são realistas, já que o grupo MSRA está realizando ambas tarefas de segmentação e classificação.

3.10 Comentários

Os resultados obtidos neste projeto mostram-se satisfatórios para imagens em contexto real. O experimento 1 comprova a complexidade do *dataset* COCO e o experimento 4 revela a importância do uso das máscaras de segmentação para a melhoraria de desempenho no classificador. Utilizar as máscaras aumenta a precisão média do classificador em 0.05 e o *recall* médio em 0.04 quando comparado ao uso somente das *bounding boxes*. As técnicas de pré-processamento empregadas a partir do experimento 2 também revelam grande impacto nos resultados. Um aumento de 0.11 na precisão média e 0.08 no *recall* médio é observado ao comparar os experimentos 1 e 2.

As máscaras de segmentação minimizam a probabilidade de ocorrência de diversos objetos contidos em uma única instância. É normal que os objetos se confundam facilmente com o background, e além disso, podem ocorrer

situações em que os *bounding boxes* se sobrescrevem, dificultando ainda mais o reconhecimento das instâncias. Como todos os *bounding boxes* se encontram no mesmo plano de imagem nas anotações do *dataset*, não existe a informação de qual objeto está se destacando em uma determinada imagem. O mesmo ocorre quando há sobreposição das máscaras de segmentação, porém com menos frequência.

O experimento 5 demonstra resultados melhores que o estado da arte no *dataset* do COCO com uma melhoria de 0.29 na precisão média. Entretanto, ressalta-se mais uma vez que a tarefa realizada por este projeto é apenas a de classificação, não incluindo a segmentação das instâncias na imagem. O estado da arte atinge um resultado de precisão melhor que o experimento 5 em 11 classes: *elephant, bus, bear, giraffe, pizza, toilet, horse, laptop, tv, dog* e *cat*.

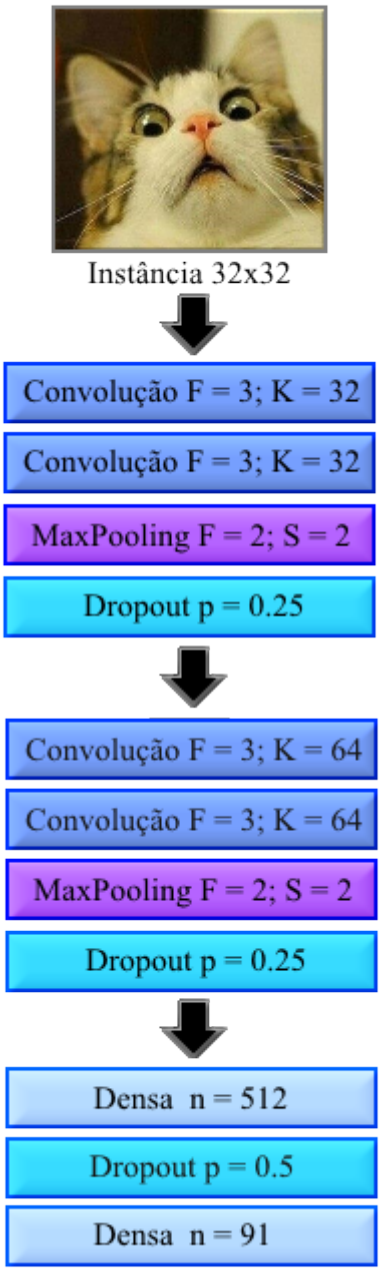

Figura 16: A arquitetura 1 apresenta uma CNN convencional com camadas convolutivas, de pooling, e densa.

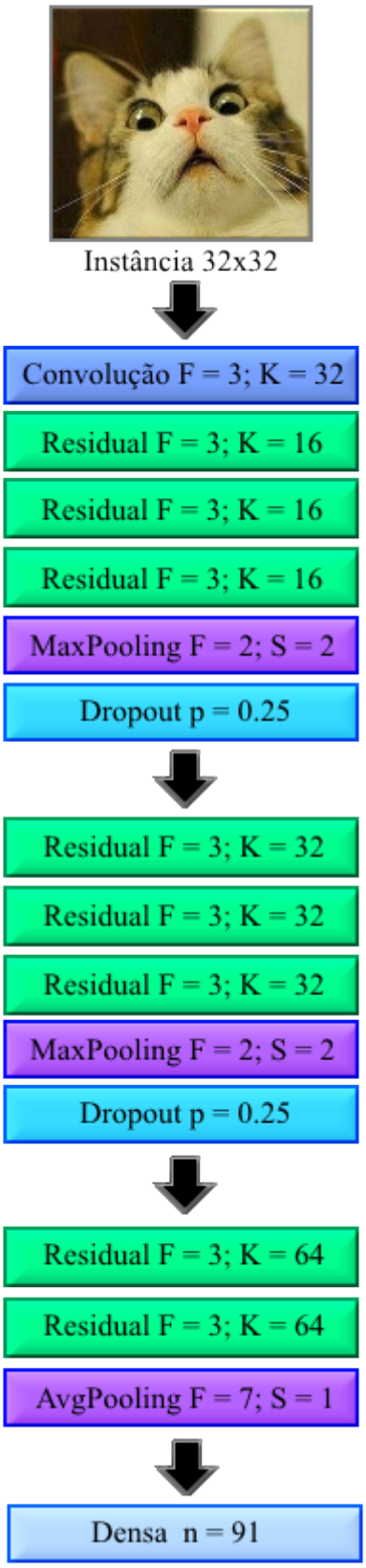

Figura 17: A arquitetura 2 apresenta uma ResNet mais complexa com camadas convolutivas e residuais.

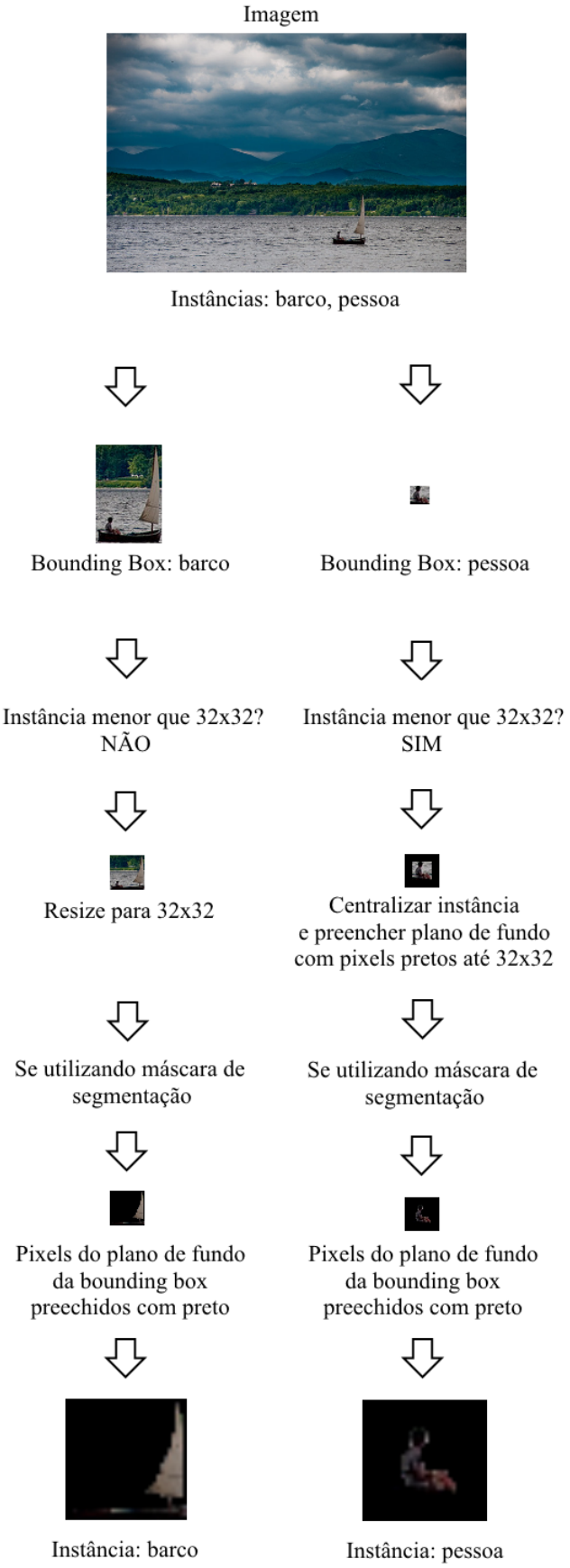

Figura 18: O pré-processamento realizado nas instâncias antes de serem alimentadas para a rede. Instâncias de tamanho maior que 32x32 são redimensionadas enquanto instâncias menores sofrem um *padding* de *pixels* pretos até 32x32.

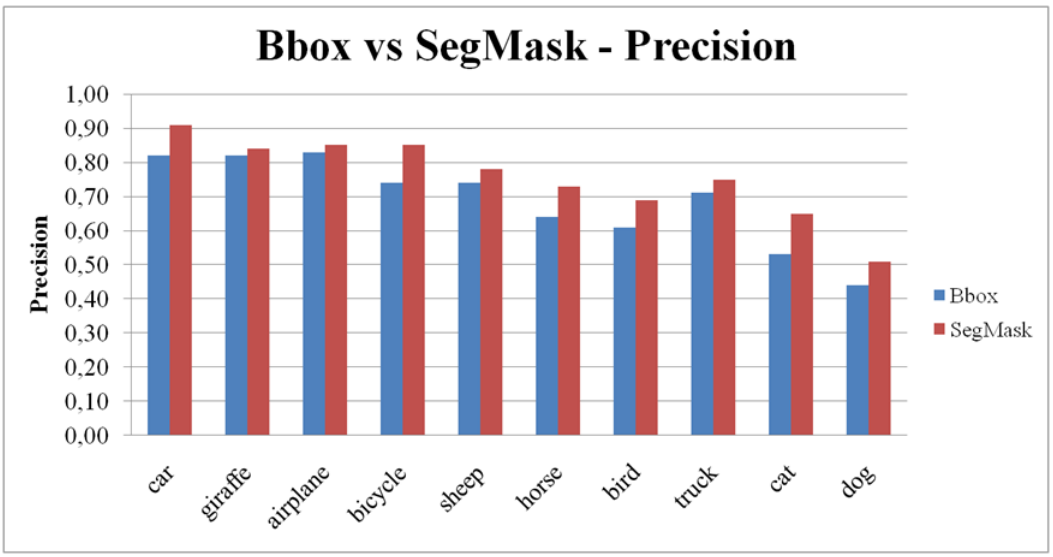

Figura 19: A diferença em precisão dos resultados obtidos utilizando os rótulos de segmentação.

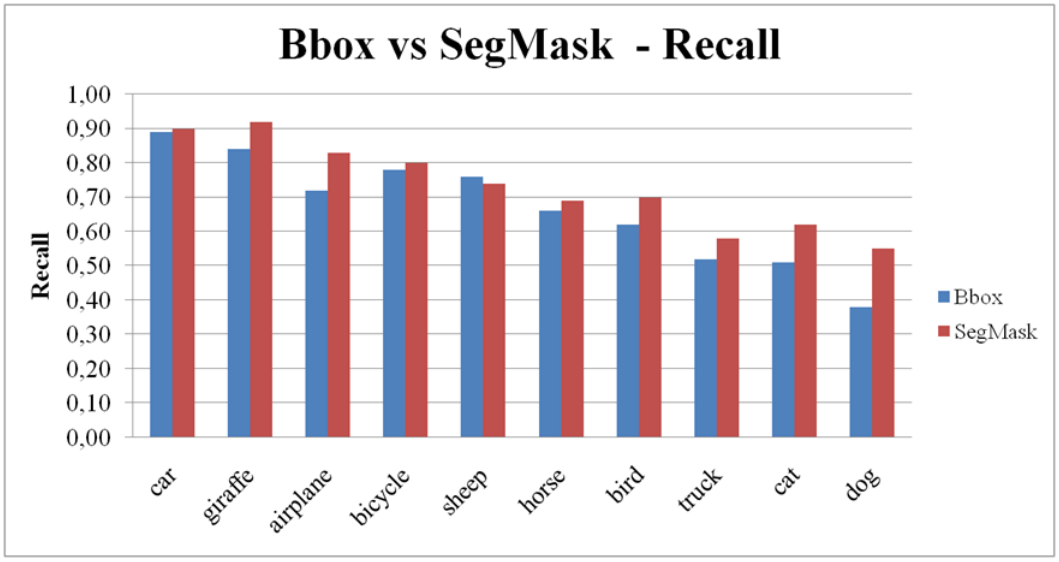

Figura 20: A diferença em *recall* dos resultados obtidos utilizando os rótulos de segmentação.

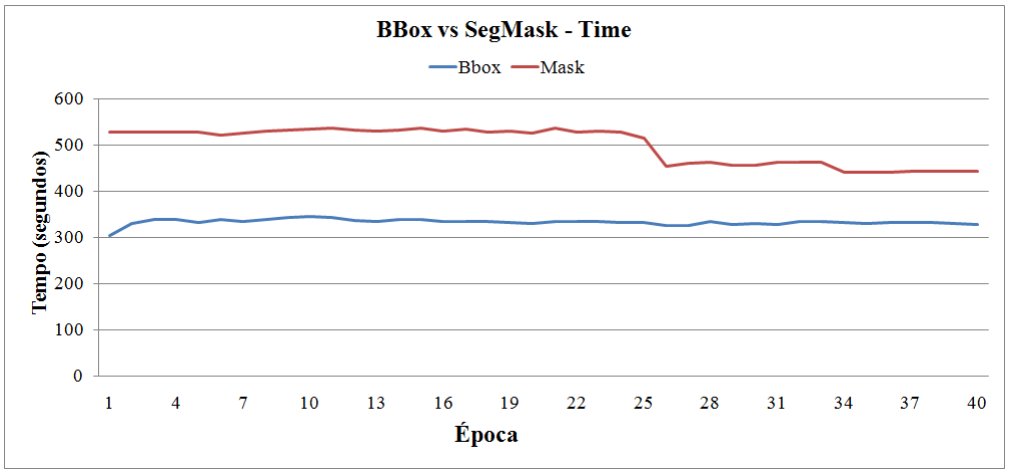

Figura 21: A diferença em tempo de execução dos resultados obtidos utilizando os rótulos de segmentação.

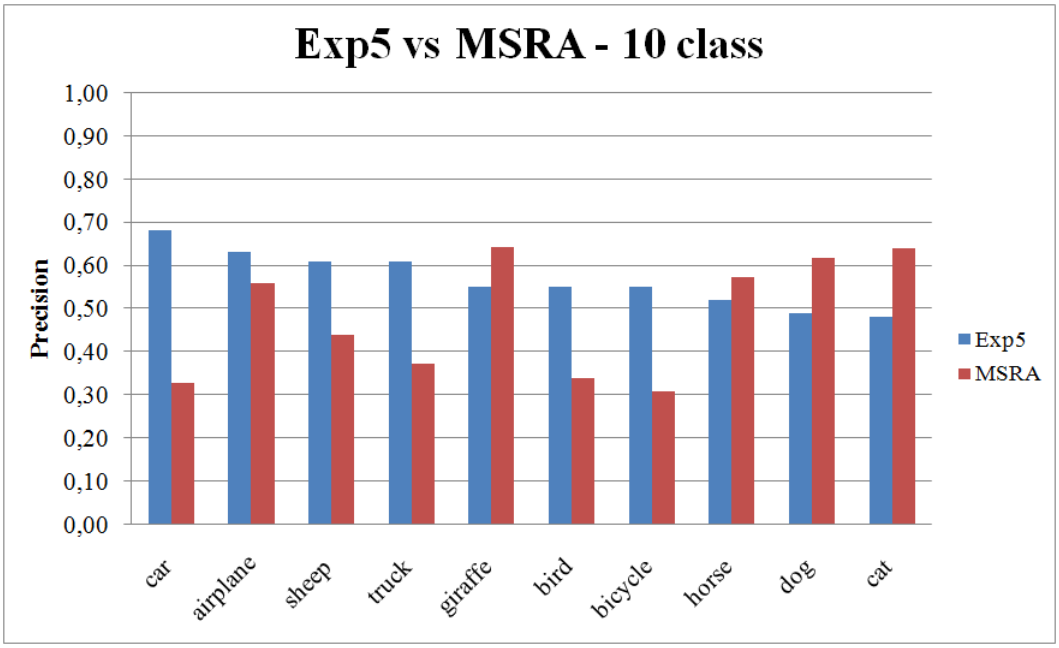

Figura 22: Comparação entre os resultados de precisão obtidos no experimento 5 e o estado da arte do grupo MSRA utilizando apenas 10 classes.

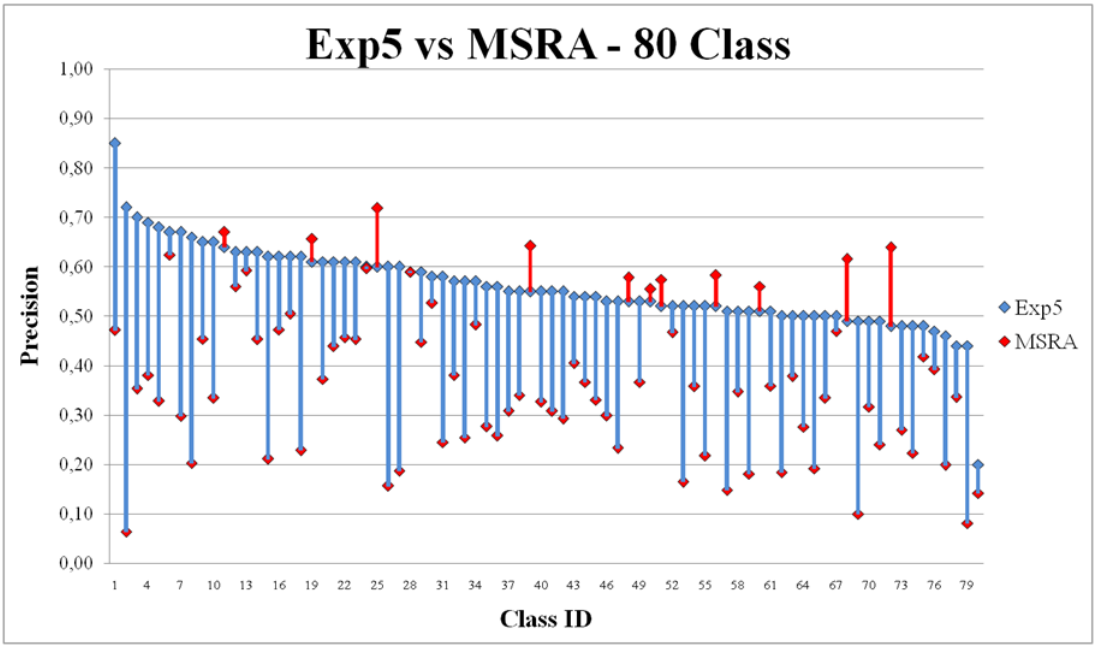

Figura 23: Comparação entre os resultados de precisão obtidos no experimento 5 e o estado da arte do grupo MSRA utilizando todas as classes do *dataset*. O ID da classe é ordenado conforme as posições das classes na tabela 7. Isso significa que o gráfico está em ordem decrescente de resultados de F_1 obtidos no experimento 5. As linhas em azul indicam diferença positiva do experimento com o estado da arte e as linhas em vermelho indicam diferença negativa.

4
Conclusões e Trabalhos Futuros

Como observamos no capítulo 3, a classificação de objetos em contexto real pode ser aprimorada com o uso de diversos recursos. O experimento 2 confirma a melhoria de desempenho com as técnicas de aumento de *dataset* empregadas. O experimento 3 comprova a importância da topologia da rede e do uso das redes residuais. E por fim, o experimento 4 demonstra a utilidade das máscaras de segmentação no desempenho da classificação, reduzindo o ruído do *background* nos objetos. Os resultados finais da classificação superam os resultados do estado da arte para o conjunto de tarefas de classificação e segmentação. Apesar disso, não é possível afirmar que o modelo do projeto é superior ao modelo do estado da arte.

O sistema de classificação de imagens em contexto real pode ser útil em diversas aplicações. Entre elas podemos citar: o reconhecimento de placas de trânsito de veículos autônomos, o reconhecimento de pedestres para auto-ajuste de sinais de trânsito, uso no entretenimento, realidade aumentada, entre outros. Como esse projeto não apresenta a tarefa de segmentação, suas utilidades passam a ser mais restritas. Entretanto, qualquer aplicação que forneça a *bounding box* como entrada pelo usuário pode utilizar desse projeto.

Esse projeto apresenta resultados promissores, entretanto, ainda possui algumas limitações. Uma dessas limitações é o fato do preditor classificar apenas um único objeto dentro da *bounding box*. Como exemplo, podemos ver na figura 18 que a instância de *bounding box* barco engloba por completo o objeto de classe pessoa. Isso significa que, caso não sejam utilizadas as máscaras de segmentação, o preditor poderia classificar essa instância como sendo da classe pessoa, o que teoricamente seria correto já que essa *bounding box* contém a pessoa, entretanto, ele estaria errado pois a anotação dessa *bounding box* é da classe barco. As máscaras de segmentação tentam reduzir esse problema, todavia, ele ainda está presente em imagens com múltiplos objetos se sobrepondo.

A segunda limitação também está relacionada com a sobreposição de objetos. A figura 24 apresenta um exemplo de sobreposição onde a instância é separada em duas partes. Essa separação dificulta a tarefa de classificação do preditor.

Uma outra limitação que pode ser mencionada é o fato do classificador utilizar as máscaras de segmentação apenas durante o treino do modelo. Essas anotações também poderiam ser utilizadas durante a etapa de predição, o que provavelmente aumentaria o desempenho do preditor. Entretanto, utilizar as máscaras de segmentação durante a predição significa restringir as utilidades do projeto.

Por fim, podemos dizer que o *dataset* do MS COCO é extremamente abrangente em suas 91 categorias de objetos em contexto real, e por conta disso, suas aplicações ficam limitadas somente à imaginação do programador.

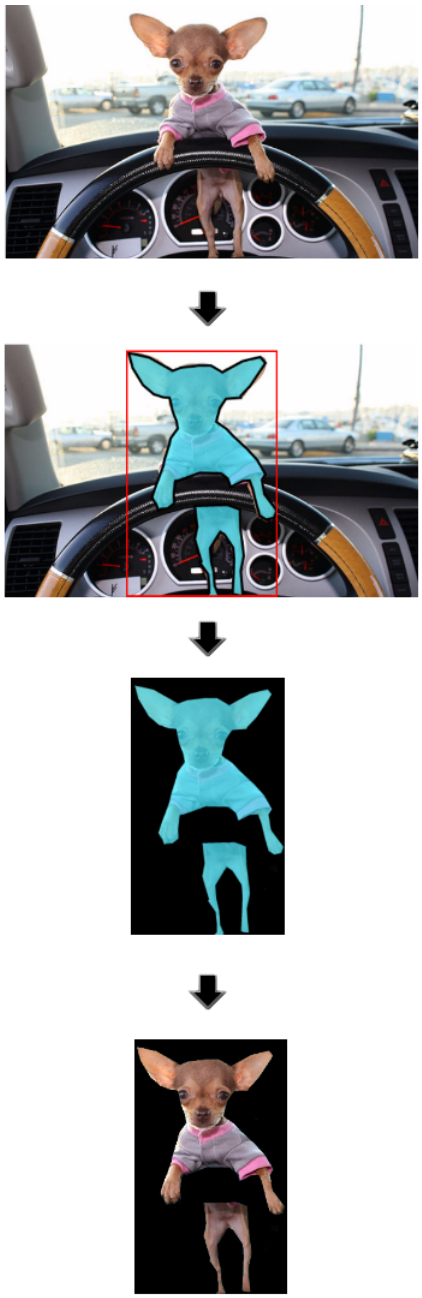

Figura 24: A sobreposição de objetos pode fazer com que as anotações fiquem segmentadas, tornando o objeto um polígono descontínuo.

Referências bibliográficas

1 AL-RFOU, R.; ALAIN, G.; ALMAHAIRI, A. ; ET AL.. **Theano: A python framework for fast computation of mathematical expressions**. CoRR, abs/1605.02688, 2016.

2 CHOLLET, F.. **Keras**. https://github.com/fchollet/keras, 2015.

3 **Cs231n convolutional neural networks for visual recognition**. http://cs231n.github.io/convolutional-networks/. Accessed: 2016-06-15.

4 EVERINGHAM, M.; ESLAMI, S. A.; VAN GOOL, L.; WILLIAMS, C. K.; WINN, J. ; ZISSERMAN, A.. **The pascal visual object classes challenge: A retrospective**. International Journal of Computer Vision, 111(1):98–136, 2015.

5 FUKUSHIMA, K.. **Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position**. Biological Cybernetics, 36:193–202, 1980.

6 GLOROT, X.; BORDES, A. ; BENGIO, Y.. **Deep sparse rectifier neural networks.** In: AISTATS, volumen 15, p. 275, 2011.

7 HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. CoRR, abs/1512.03385, 2015.

8 KRIZHEVSKY, A.; HINTON, G.. **Learning multiple layers of features from tiny images**. 2009.

9 KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E.. **Imagenet classification with deep convolutional neural networks**. In: Pereira, F.; Burges, C. J. C.; Bottou, L. ; Weinberger, K. Q., editors, ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 25, p. 1097–1105. Curran Associates, Inc., 2012.

10 LECUN, Y.; BOTTOU, L.; BENGIO, Y. ; HAFFNER, P.. **Gradient-based learning applied to document recognition**. In: Haykin, S.; Kosko, B., editors, INTELLIGENT SIGNAL PROCESSING, p. 306–351. IEEE Press, 2001.

11 LEE, C.-Y.; XIE, S.; GALLAGHER, P.; ZHANG, Z. ; TU, Z.. **Deeply-supervised nets.** In: AISTATS, volumen 2, p. 6, 2015.

12 LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P. ; ZITNICK, C. L.. **Microsoft COCO: common objects in context**. CoRR, abs/1405.0312, 2014.

13 LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P. ; ZITNICK, C. L.. **Microsoft COCO: metrics**. `http://mscoco.org/dataset/#detections-eval`, 2015.

14 MCCULLOCH, W. S.; PITTS, W.. **A logical calculus of the ideas immanent in nervous activity**. The bulletin of mathematical biophysics, 5(4):115–133, 1943.

15 NAIR, V.; HINTON, G. E.. **Rectified linear units improve restricted boltzmann machines**. In: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML-10), p. 807–814, 2010.

16 RUMELHART, D.; HINTONT, G. ; WILLIAMS, R.. **Learning representations by back-propagating errors**. Nature, 323(6088):533–536, 1986.

17 SRIVASTAVA, N.; HINTON, G. E.; KRIZHEVSKY, A.; SUTSKEVER, I. ; SALAKHUTDINOV, R.. **Dropout: a simple way to prevent neural networks from overfitting.** Journal of Machine Learning Research, 15(1):1929–1958, 2014.

18 TURING, A. M.. **On computable numbers, with an application to the entscheidungsproblem**. J. of Math, 58(345-363):5, 1936.

19 VON AHN, L.; BLUM, M.; HOPPER, N. J. ; LANGFORD, J.. **Captcha: Using hard ai problems for security**. In: INTERNATIONAL CONFERENCE ON THE THEORY AND APPLICATIONS OF CRYPTOGRAPHIC TECHNIQUES, p. 294–311. Springer, 2003.

20 WERBOS, P.. **Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences**. PhD thesis, Harvard University, Cambridge, MA, 1974.

21 WITTEN, I. H.; FRANK, E.; HALL, M. A. ; PAL, C. J.. **Data Mining: Practical machine learning tools and techniques**. Morgan Kaufmann, 2016.

22 ZEILER, M. D.. **Adadelta: an adaptive learning rate method**. arXiv preprint arXiv:1212.5701, 2012.