PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Bacharelado em Engenharia de Computação

André Dias dos Santos Júnior

LOCALIZAÇÃO E CLASSIFICAÇÃO DE OBJETOS USANDO REDES NEURAIS CONVOLUTIVAS DENSAS

Belo Horizonte

25 de Novembro de 2020

André Dias dos Santos Júnior

LOCALIZAÇÃO E CLASSIFICAÇÃO DE OBJETOS USANDO REDES NEURAIS CONVOLUTIVAS DENSAS

Monografia apresentado ao Curso de Bacharelado em Engenharia de Computação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Dr. Zenilton Kléber Gonçalves do Patrocínio Júnior

Belo Horizonte

25 de Novembro de 2020

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

André Dias dos Santos Júnior

LOCALIZAÇÃO E CLASSIFICAÇÃO DE OBJETOS USANDO REDES NEURAIS CONVOLUTIVAS DENSAS

Monografia apresentado ao Curso de Bacharelado em Engenharia de Computação da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Prof. Dr. Zenilton Kléber Gonçalves do Patrocínio Júnior – PUC Minas

---

Prof. Dr. Luiz Carlos Figueiredo – PUC Minas

Prof. Leonardo Vilela Cardoso – PUC Minas

Belo Horizonte, 25 de Novembro de 2020.

À minha amada avó, que sempre me apoiou.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, que me deu forças para prosseguir, mediante aos obstáculos.

Agradeço também à minha avó por ter me apoiado incondicionalmente e por ter sempre acreditado em mim.

Agradeço também à minha mãe por ser um suporte sólido e por sempre me apoiar quando precisei.

Agradeço também ao meu orientador, o professor Zenílton pelo apoio, a presença, o auxílio prestado e pelo aprendizado ao longo da minha jornada.

Agradeço ao meu pai por torcer por mim e pelas orações.

E agradeço à todos os familiares e amigos que, de alguma forma me auxiliaram ao longo deste caminho.

"Quando eu era menino, falava como menino, sentia como menino, discorria como menino, mas, logo que cheguei a ser homem, acabei com as coisas de menino."

I Coríntios 13:11

RESUMO

Atualmente há um expressivo crescimento no volume de conteúdo visual, como imagens e vídeos, disponíveis para utilização. As informações contidas em imagens e vídeos são utilizadas pelas mais variadas áreas da sociedade, como médica, industrial e, até mesmo, para fins pessoais. Todos os dias são estudados novos métodos computacionais para fazer recuperação e análise de imagens. Dois métodos que tem sido frequentemente estudados em conjunto são a localização e classificação de objetos. Arquiteturas de *Deep Learning* podem ser implementadas para fazer localização e classificação de objetos, obtendo bons resultados, porém possuem certas limitações como o tamanho dos objetos ou a resolução da imagem. Para contornar essa limitação, pretende-se acrescentar camadas de convolução ao final de uma rede convolutiva densa para fazer a localização dos objetos de diversos tamanhos de forma mais apurada.

Palavras-chave: *Deep Learning*, Localização, Classificação, Convolução.

ABSTRACT

Currently, there is an expressive growth in the amount of visual content, such as images and videos available for use. The information contained in images and videos are used by the most areas of society, such as medical, industry and even for personal purposes.Every day new computational methods for recovering and analyzing images are studied. Two methods that usually been studied together are the localization and classification of objects. Architectures of Deep Learning can be implemented to localizate and classify objects, obtaining good results, however they have certain limitations such as the size of the objects or the resolution of the image. To avoid this limitation, it is intended to add convolution layers in the end of a Dense Convolutional Network to make the location of objects of different sizes more accurately.

Keywords: Deep Learning, Localization, Classification, Convolution.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CNN – *Convolutional Neural Networks* - Redes Neurais Convolucionais

CVPR – *Conference on Computer Vision and Pattern Recognition* - Conferência de Visão Computacional e Reconhecimento de Padrões

DBN – *Deep Belief Networks*

DenseNet – *Densely Connected Convolutional Networks* - Redes Convolutivas Densamente conectadas

DSSD – *Deconvolutional Single-Shot Detector*

FCN – *Fully Convolutional Network* - Rede Completamente Convolucional

FPS – *Frames Per Second* - Quadros Por Segundo

ILSVRC – *ImageNet Large Scale Visual Recognition Challenge*

mAP – *Mean Average Precision* - Média das precisões Médias

MLP – *Multi-Layer Perceptron*

MSE – *Mean Squared Error* - erro quadrático médio

PASCAL VOC – *PASCAL Visual Object Classes*

RBM – *Restricted Boltzmann Machines*

R-CNN – *Regions with CNN features*

ReLU – *Rectified Linear Unit* - Ativação Linear Retificada

ResNet – *Residual Network* - Rede Neural Residual

RNA – Rede Neural Artificial

SSD – *Single-Shot multi-box Detector*

YOLO – *You Only Look Once* - você só olha uma vez

SUMÁRIO

1 INTRODUÇÃO

Atualmente há uma produção massiva de conteúdo visual, como imagens e vídeos, disponíveis para utilização. Podemos vincular este fato à popularização das tecnologias produtoras destes tipos de conteúdo, como celulares com câmeras, bem como a expansão da Internet e seus canais de comunicação, que utilizam de imagens e vídeos para divulgação de informações. As informações contidas em imagens e vídeos são utilizadas pelas mais variadas áreas da sociedade, como médica, industrial e, até mesmo, para fins pessoais e de entretenimento.

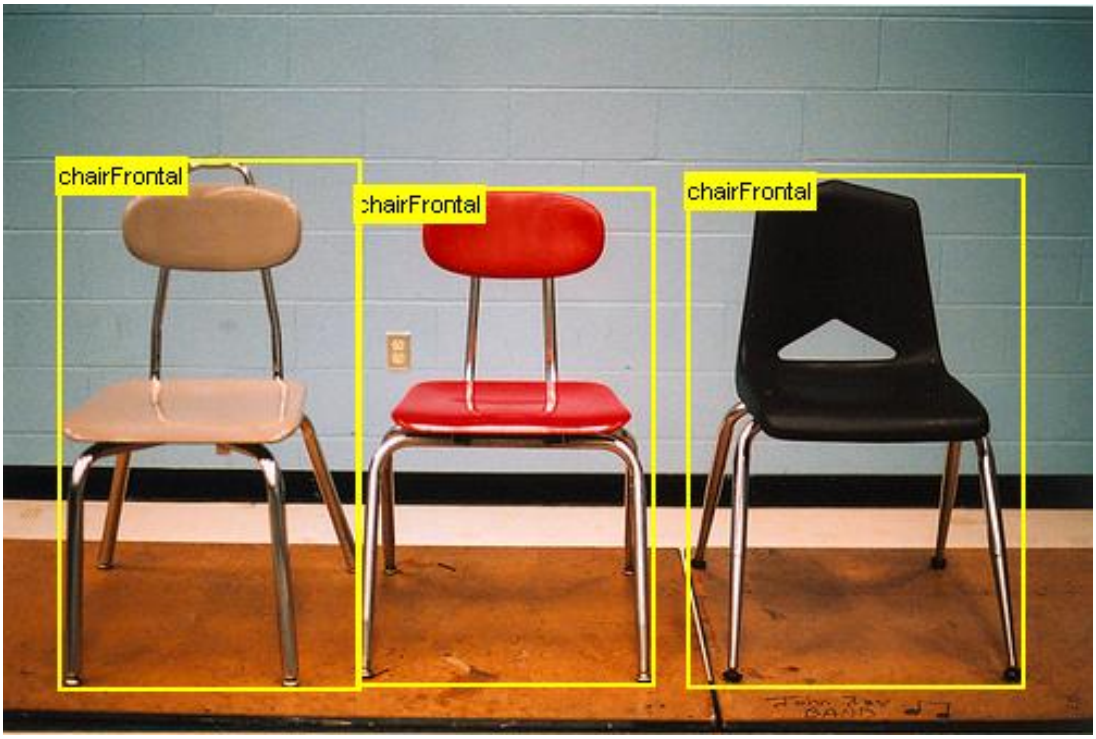
O processamento digital de imagens contribui para a descoberta de informação visual contida em imagens e vídeos. Um dos problemas de processamento digital de imagens amplamente explorados na literatura é o problema de classificação e localização de objetos em imagens. Everingham et al. (2015) definem que o problema de classificação consiste em responder para cada classe de objeto se existe ou não uma ou mais instâncias daquele objeto na imagem e, o problema de localização consiste em dizer onde na imagem estão as instâncias dos objetos reconhecidos pelo classificador.

A Figura 1 mostra exemplos de como devem ser as saídas de um algoritmo de classificação e localização de objetos. Os retângulos destacados em torno dos objetos são resultados do algoritmo de localização que determina que dentro daquela região existe um objeto de interesse e os rótulos destacados em cima dos retângulos são os resultados do algoritmo de classificação, que determina que o objeto contido dentro da região pertence àquela classe.

O uso das *Convolutional Neural Networks* - Redes Neurais Convolucionais (CNN) tem se tornado cada vez mais populares, principalmente em alguns dos problemas clássicos de processamento de imagens. A primeira abordagem desse tipo foi proposta por Fukushima (1980) para fazer reconhecimento de caracteres escritos a mão. Mais tarde, Lecun et al. (1998) desenvolveram uma arquitetura de CNN para a mesma tarefa e obtiveram uma acurácia de 70%, sendo essa muito superior aos resultados obtidos por qualquer outro classificador utilizado até então.

Contudo, as CNN só passaram a ser utilizadas com mais frequência algum tempo depois. Deng et al. (2009) lançaram o ImageNet, uma base de dados de larga escala com mais de 14 milhões de imagens anotadas manualmente divididas em mais de 22 mil classes diferentes. Essa base de imagens pode ser usada para as tarefas de classificação, detecção e localização de objetos. Além disso, Deng et al. (2009) também lançaram o *ImageNet*

Figura 1 – Exemplo de Classificação e Localização de Objetos



Fonte: Everingham et al. (2015).

Large Scale Visual Recognition Challenge (ILSVRC), uma competição global anual, aonde os competidores são avaliados nas tarefas de classificação de imagens, detecção de objetos e localização de objetos. Krizhevsky, Sutskever e Hinton (2012) alcançaram o melhor resultado no ILSVRC de 2012 usando a AlexNet - uma CNN - e desde então, as CNNs obtém sempre os melhores resultados nos desafios.

Uma outra área impactada pelo uso das CNNs é a área de segmentação semântica. Long, Shelhamer e Darrell (2014) propuseram o uso de uma *Fully Convolutional Network* - Rede Completamente Convolucional (FCN) para fazer segmentação semântica e conseguiram obter resultados superiores ao estado da arte, utilizando camadas de deconvolução para aumentar a resolução dos mapas de filtros gerados na saída da CNN. Noh, Hong e Han (2015) propuseram um trabalho mais avançado de forma a tratar algumas limitações encontradas por Long, Shelhamer e Darrell (2014). Essa arquitetura funciona com a estrutura de um *autoencoder*, usando as camadas de deconvolução que não servem para aumentar a resolução, mas para recuperar as informações da imagem de forma mais refinada, de forma a melhorar os resultados da segmentação.

Nas tarefas de localização de objetos as CNNs também têm se destacado com bons resultados. Redmon et al. (2015) propuseram o *You Only Look Once* - você só olha uma vez (YOLO) e obtiveram um bom resultado fazendo localização e classificação de objetos em tempo real, conseguindo processar 45 *Frames Per Second* - Quadros Por Segundo (FPS) com uma *Mean Average Precision* - Média das precisões Médias (mAP) de

63,4%. Ren et al. (2017) propuseram o *Regions with CNN features* (R-CNN) e conseguiram obter resultados significativamente melhores para as tarefas de localização e classificação chegando a uma mAP de até 78,8%. Numa implementação alternativa com o enfoque no processamento em tempo real, a mAP cai um pouco, chegando a 73,2% porém, ele processa apenas 7 FPS.

Liu et al. (2015) propuseram o *Single-Shot multi-box Detector* (SSD), uma abordagem que além de obter uma mAP elevada (74,3%), supera a velocidade alcançada no YOLO atingindo 59 FPS. Esses resultados são especialmente relevantes, pois, além de serem competitivos com o estado da arte, trabalham com imagens menores - enquanto os quadros processados pelo YOLO têm dimensões 448×448 e os processados pelo R-CNN têm 1000×600, o SSD processa quadros de dimensões 300×300. O principal ganho em trabalhar com imagens menores é diminuir o consumo de memória e ter um método mais robusto por poder obter resultados competitivos mesmo que com imagens menores. Por um outro lado, trabalhar com imagens menores pode significar que o método seja mais lento ao processar imagens maiores.

Por fim, Fu et al. (2017) propuseram uma abordagem alternativa na resolução dos problemas de localização e classificação de objetos utilizando camadas de deconvolução a *Deconvolutional Single-Shot Detector* (DSSD). A DSSD recebe essa sigla, pois é uma extensão do SSD (LIU et al., 2015). A diferença é que, ao final, ele acrescenta camadas de deconvolução e, com os resultados das deconvoluções ele faz a classificação e a localização. Os resultados obtidos superam os apresentados pelo SSD em mAP, alcançando 81,5% embora, o DSSD perde na velocidade (13,6 FPS).

1.1 Motivação

Estudos envolvendo CNNs foram realizados recentemente em áreas como classificação de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; SIMONYAN; ZISSERMAN, 2014), classificação e detecção de objetos (FU et al., 2017; LIN et al., 2014), segmentação semântica de imagens (LONG; SHELHAMER; DARRELL, 2014; Noh; Hong; Han, 2015), segmentação de objetos em vídeos (Caelles et al., 2017; VOIGTLAENDER; LEIBE, 2017). Isso se deve aos bons resultados obtidos pelos métodos aplicados nas respectivas áreas.

Além disso, o problema de classificação e detecção de objetos é aproveitado como solução para problemas mais específicos, como detecção facial (Yang; Jiachun, 2018), contagem de pessoas (Ren; Fang; Djahel, 2017) e detecção de pedestres (Lan et al., 2018).

Por fim o uso de camadas adicionais de convolução ao final de uma CNN pode gerar bons resultados na localização e classificação de objetos como apresentado por Liu et al.

(2015) e Fu et al. (2017). Em suma, pode-se dizer que o tema ainda tem muito que ser explorado, que a proposta é promissora, e que bons resultados podem trazer contribuições para trabalhos futuros.

1.2 Objetivos

O objetivo do trabalho é implementar modificações na arquitetura da *Densely Connected Convolutional Networks* - Redes Convolutivas Densamente conectadas (DenseNet)121 de forma a possibilitar que ela faça localização e classificação de objetos, conforme proposto por Liu et al. (2015) e por Huang et al. (2017).

1.2.1 Objetivos Específicos

Para que o objetivo geral do trabalho seja alcançado, será necessário cumprir os seguintes objetivos:

- Fazer um estudo para entender o funcionamento dos algoritmos de Localização e classificação de objetos;
- Levantar quais bases de dados serão utilizadas para fazer o treinamento e a avaliação do modelo proposto. Além disso, também levantar qual ou quais métricas serão utilizadas para avaliar os resultados;
- Implementar uma Rede Neural Convolutiva Densa modificada com os módulos de predição para localização e classificação propostos por Liu et al. (2015) e Huang et al. (2017);
- Fazer o treinamento do modelo desenvolvido com as bases selecionadas. Após o treinamento, fazer os testes e registrar os resultados;
- Analisar os resultados, comparando-os com a literatura.

1.3 Justificativa

Com os avanços tecnológicos alcançados ao longo das últimas décadas, a produção de conteúdo multimídia tem crescido consideravelmente e tem sido amplamente explorada pelos mais diversos setores da sociedade. Todos os dias mais de 95 milhões de fotos e vídeos são postadas no Instagram*. Esse imenso volume de dados que são produzidos e

*https://www.instagram.com

acessados em multimídia inviabiliza a manipulação deste conteúdo por meio de ação humana; criando assim, a necessidade de automatizar a recuperação e análise de informações relevantes contidas nas mesmas.

Nesse sentido, todos os dias são estudados novos algoritmos e novas técnicas para fazer recuperação de informação multimídia. E uma das formas amplamente utilizadas é a de localização e classificação de objetos em imagens. Everingham et al. (2015) definem que o problema de classificação consiste em responder para cada classe de objeto se existe ou não uma ou mais instâncias daquele objeto na imagem e, o problema de localização consiste em dizer onde na imagem estão as instâncias dos objetos reconhecidos pelo classificador.

É válido estabelecer que "[...]os mecanismos sensitivos dos seres humanos (como visão e audição) sugerem a necessidade de uma arquitetura profunda para extrair a sua estrutura complexa." (DENG; YU, 2014). Porém, de acordo com Caelles et al. (2017), uma grande limitação ao utilizar arquiteturas de *Deep Learning* é a necessidade de treinamento com um grande volume de dados. Essa limitação torna o processo mais custoso em termos de tempo de processamento e outros recursos computacionais(como memória). Além disso, a base de dados deve ter os resultados da classificação e localização esperados feitos de forma manual, o que aumenta o esforço humano.

Tendo isso em vista, a proposta de utilizar a DenseNet (Huang et al., 2017) pré-treinada em classificação de imagens visa diminuir o custo por meio do *Transfer Learning*. Além disso, utilizando as camadas adicionais de convolução como proposto por Liu et al. (2015), espera-se obter resultados ainda melhores. Esses resultados melhores repercutiriam de forma positiva em outras áreas relacionadas à localização e classificação de objetos.

1.4 Organização do texto

O Capítulo 2 traz os principais conceitos relacionados à *Deep Learning*, redes neurais artificiais, classificação de objetos em imagens e detecção de objetos em imagens, necessários à compreensão do trabalho. O Capítulo 3 contém um resumo dos trabalhos relacionados à monografia. O Capítulo 4 contém a metodologia a ser seguida para o desenvolvimento do trabalho. O Capítulo 5 descreve os experimentos realizados e os resultados obtidos. O Capítulo 6 apresenta a conclusão do trabalho com base nos resultados obtidos e apresenta propostas de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão apresentados os principais conceitos e definições da literatura sobre Rede Neural Artificial (RNA), Deep Learning, classificação e localização de objetos.

2.1 *Machine Learning*

Machine Learning ou Aprendizado de Máquina é um campo de estudo cada vez mais recorrente na área de tecnologia. Em uma definição alternativa, pode-se dizer que "[...]Aprendizado de Máquina é uma área de Inteligência Artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática." (MONARD; BARANAUSKAS, 2003). Entre as principais aplicações de Aprendizado de máquina, pode se destacar o reconhecimento de padrões, classificação de imagens e a mineração de dados.

Monard e Baranauskas (2003) definem ainda que o aprendizado de máquina pode ser supervisionado ou não-supervisionado. Quando falamos do primeiro caso, queremos dizer que o algoritmo recebe um conjunto de dados na entrada, processa esses dados e retorna uma saída. Essa saída é comparada com um valor previamente associado à entrada, comumente conhecido como rótulo. Já no aprendizado não-supervisionado, as entradas não possuem rótulo algum, cabendo ao algoritmo fazer a distinção das diversas entradas. Geralmente, nesses casos é necessária uma análise posterior à execução do algoritmo para se entender os resultados obtidos.

O aprendizado supervisionado se divide em dois grupos menores: classificação e regressão. Santos (2012) define que quando o problema possui um conjunto discreto de saídas e o objetivo é atribuir a qual dessas saídas a entrada pertence, se trata de um problema de classificação. Santos (2012) define ainda que quando o objetivo é prever uma saída de valor contínuo pra uma entrada, se trata de um problema de regressão.

As seções 2.1.1 e 2.1.2 irão definir melhor esses conceitos.

2.1.1 Regressão

Como definido anteriormente, o problema de regressão consiste em calcular para cada entrada uma saída que possua um valor contínuo. Dosualdo (2003) define que a regressão consiste em fazer uma relação entre os atributos $X = x_1, x_2, ..., x_n$ - onde X é o conjunto de entrada e cada x_i é um atributo numérico ou quantitativo - e Y, tal que

Y é um atributo ou um conjunto de atributos meta. Apté e Weiss (1997) definem essa relação por meio da Equação 2.1.

$$Y = f(x_1, x_2, ..., x_n) \tag{2.1}$$

A relação entre o(s) atributo(s) X e o(s) atributo(s)-meta Y alcançada como resultado de um algoritmo de regressão é chamado de modelo. Após a definição do modelo é importante fazer uma avaliação para dizer o quão confiável será a predição gerada pelo modelo. Existem diversas funções usadas na avaliação dos algoritmos de regressão, e uma das mais comuns é o *Mean Squared Error* - erro quadrático médio (MSE) que é definido pela Equação 2.2.

$$E = \frac{1}{N}\sum_{i=0}^{N}(Y_i - f(X)_i)^2 \tag{2.2}$$

Em que E é o erro, N é o tamanho do conjunto de amostras, Y_i é o valor esperado do atributo-meta e $f(X_i)$ é o resultado do modelo para a entrada X_i. O objetivo do treinamento da predição, é minimizar esse erro quadrático, de forma que - em tese - o resultado previsto seja próximo do esperado.

Alguns exemplos de problemas de regressão são:

1. Predizer o percentual de gordura que uma pessoa possui no corpo, recebendo como entrada atributos como altura, idade, peso, sexo (DOSUALDO, 2003);
2. Predizer o preço de um imóvel baseado em atributos como o tamanho, número de cômodos, número de quartos, etc. (PEREIRA; GARSON; ARAÚJO, 2012).

Uma grande limitação nos métodos de regressão é que eles em sua maioria solucionam problemas que possuem apenas um atributo-meta. Porém, os métodos de regressão baseados em RNA podem retornar múltiplos resultados. Esse termo será definido na Sessão 2.2.

2.1.2 Classificação

Como mencionado anteriormente, o problema de classificação consiste em fazer uma predição quando as saídas são discretas. A ideia dos algoritmos de classificação é definir para cada entrada, um rótulo ou classe no meio de um conjunto finito de rótulos. Um método estatístico muito utilizado nos problemas de classificação é a regressão logística. A regressão logística é bem similar à regressão linear, com a diferença que o resultado é um número entre zero e um, configurando na verdade, a probabilidade de, dado

uma entrada X tal que $X = x_1, x_2, ..., x_n$ gerar uma saída verdadeira para a hipótese Y. Normalmente, os modelos de regressão logística aplicam a função sigmóide, descrita pela Equação 2.3.

$$Y = \frac{1}{1 + e^{-g(X)}} \tag{2.3}$$

A regressão logística se ajusta muito bem a problemas de classificação binária. Porém, a grande maioria dos problemas de classificação possuem mais de uma classe, consistindo então em dizer para uma entrada X tal que $X = [x_1, x_2, ..., x_n]$ a qual classe de Y ela pertence, tal que $Y = [y_1, y_2, ..., y_m]$. Do e Poulet (2015) afirmam que as principais abordagens para solucionar problemas de classificação com M classes consistem em dividir o problema em vários problemas de classificação binária.

Essa divisão pode ser feita de duas formas diferentes:

- A primeira consiste em gerar $M-1$ problemas de classificação binária e resolver cada um de forma independente. Após resolver os problemas de classificação, determinar a classe final a partir das probabilidades obtidas selecionando a classe que obteve a maior probabilidade. Essa abordagem é conhecida como *one-versus-all*.

- A segunda consiste em combinar as classes em pares e, para cada par de classe gerar um problema de classificação. Isto é, seja Y um conjunto de classes, para cada classe $y \in Y$ criar um problema de classificação para y e cada y_β tal que $y_\beta \in Y - [y]$. Isso gera um conjunto de $\frac{M(M-1)}{2}$ problemas de classificação. Após gerar e solucionar os problemas, o resultado será a classe que for mais vezes selecionada.

Com base na primeira abordagem, começou a ser utilizada uma função para os problemas de classificação que envolvem múltiplas classes. Essa função é a *softmax* que calcula o valor de cada saída e divide pela soma das saídas, gerando assim uma saída normalizada que corresponde à probabilidade de o dado de entrada estar na classe. Atualmente é a função mais utilizada para problemas de classificação com múltiplas classes e é descrita pela Equação 2.4.

$$f(x)_i = \frac{e^{\beta x_i}}{\sum_{j=1}^{K} e^{\beta x_j}} \forall i = 1, ..., K ex = x_1, ..., x_k \in X^K \tag{2.4}$$

Assim como na regressão linear, a perda na regressão logística pode ser calculado pelo MSE. Porém, com o tempo, surgiram outras funções que calculam a perda de forma mais eficiente em problemas de classificação.

2.2 Redes Neurais Artificiais

As RNAs são uma das principais técnicas de aprendizagem de máquina e podem ser implementadas tanto para problemas supervisionados, quanto não-supervisionados. Jost (2015) definiu as RNAs da seguinte forma:

> As RNAs possuem inspiração nas redes neurais biológicas, constituídas de neurônios separados por camadas, que processam informações e estão conectados via pesos sinápticos, sendo na maioria das vezes sistemas adaptativos que modificam sua estrutura através de informações, que fluem pela rede durante a etapa de aprendizado (JOST, 2015).

Além disso, é importante mencionar que em uma rede neural, existem duas camadas em particular que são muito importantes: a primeira camada, que é a camada de entrada de dados, e a última camada que é a camada de saída. Existem dois tipos de RNAs: as redes *feed forward*, que processa apenas o estado atual das entradas, e as redes neurais recorrentes que processa o estado atual e o estado anterior. As redes neurais recorrentes podem ser usadas de forma eficiente para processar vídeos, séries temporais e outros dados que possuam múltiplos estados.

A RNA *feed forward* mais básica que existe é a *Perceptron*, composta apenas por um conjunto de neurônios de entrada e um único neurônio de saída. A Equação 2.5 representa a fórmula de um *Perceptron*, onde n é o número de entradas, W_i são os respectivos pesos de cada entrada, X_i são as entradas e B é o viés do neurônio.

$$Y = \left(\sum_{i=1}^{n} (W_i \times X_i) \right) - B \tag{2.5}$$

Uma rede percepron não é uma arquitetura de redes neurais artificiais muito poderosa, mas deu base para outras arquiteturas mais robustas. A principal delas é o *Perceptron* Multi-Camadas.

2.2.1 Multi-Layer Perceptron

As redes *Perceptron* Multicamadas ou *Multi-Layer Perceptron* (MLP) são baseadas nos *Perceptron*s simples, como mencionado anteriormente. A diferença, porém, é que elas trabalham com mais camadas do que simplesmente as camadas de entrada e saída. Geralmente elas possuem uma ou duas camadas intermediárias às camadas externas. Essas camadas intermediárias, também são chamadas de camadas ocultas e servem para conferir uma robustez maior ao método.

Nas MLPs, uma outra diferença é que as funções de ativação podem variar. As mais comuns são o *Rectified Linear Unit* - Ativação Linear Retificada (ReLU) (definido

pela Equação 2.6), a função sigmóide (descrita na Seção 2.1.2) e a tangente hiperbólica (definida pela Equação 2.7).

$$Y(x) = max(0, x) \tag{2.6}$$

$$Y(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.7}$$

Como foi dito anteriormente, a MLP trabalha com aprendizado supervisionado, isso quer dizer que os dados que ela opera são rotulados, e tem uma saída esperada. Quando as entradas são processadas e os resultados das saídas são calculados, eles são comparados com os valores esperados pelos rótulos e é gerado a função de erro quadrático, definida na Seção 2.1.1.

Sabendo-se disso, o objetivo para melhorar a precisão da MLP é minimizar o valor da função erro, ou seja, tornar as saídas da rede o mais próximo possível das saídas esperadas. Para tanto, é utilizado o método de retropropagação de erro que reajusta os pesos da rede de acordo com os valores obtidos usando a descida de gradiente.

Arnold et al. (2011) definem que aumentar o número de camadas em um MLP não garante uma melhoria dos resultados, pois a descida de gradiente pode chegar a um mínimo local. Além disso, o aumento do número de camadas implica em um tempo muito maior de processamento. Para lidar com esse problema surge o Deep Learning, uma arquitetura avançada com múltiplas camadas, que soluciona a dificuldade que as redes neurais possuem ao lidar com dados de alta dimensionalidade (ARNOLD et al., 2011).

2.3 *Deep Learning*

A principio não parecia ser viável manipular arquiteturas profundas de redes neurais. Porém, de acordo com Deng e Yu (2014), surgiu um algoritmo de aprendizado não-supervisionado que conseguiu aliviar empiricamente as dificuldades de otimização em arquiteturas profundas. Esse algoritmo é a *Deep Belief Networks* (DBN), um modelo generativo profundo composto de uma camada visível e várias camadas ocultas compostas por uma pilha de *Restricted Boltzmann Machines* (RBM)s.

Deng e Yu (2014) definem ainda que o aprendizado nas DBNs é feito por um algoritmo guloso que ajusta os pesos camada-por-camada com uma complexidade linear ao tamanho e à profundidade da rede. E uma relação inesperada entre as DBNs e as MLPs surgiu quando descobriu-se que ao utilizar os pesos de uma DBN com arquitetura correspondente, você consegue inicializar os pesos de uma MLP de forma a produzir melhores resultados do que utilizando pesos aleatórios.

Uma segunda alternativa para trabalhar com aumento de camadas é o empilhamento de auto-codificadores. O empilhamento de auto-codificadores consiste basicamente em inserir na saída de uma rede neural uma segunda rede neural que para cada entrada produz uma saída específica e retreinar a nova rede neural utilizando o algoritmo de retropropagação de erro (DENG; YU, 2014).

2.3.1 Redes Neurais Convolucionais

As CNN são um modelo específico de *Deep Learning* muito utilizados em aplicações de visão computacional e aplicações de reconhecimento de imagens. De acordo com Ferreira (2017), as CNNs utilizam matrizes para processar as entradas de dados, sendo essas matrizes unidimensionais para sinais e sequências, bidimensionais para imagens e tridimensionais para imagens volumétricas e vídeos.

Ao contrário das RNAs tradicionais (como MLP), as CNNs não necessariamente ligam todos os neurônios na camada de origem a todos os neurônios da camada de destino. Ferreira (2017) define que existem três tipos comuns de camadas utilizadas nas redes neurais convolucionais: as camadas de convolução, camadas de *pooling* e camadas completamente conexas.

A ideia da camada de convolução é de que cada neurônio recebe como entrada neurônios próximos e tem por objetivo criar mapas e filtros. Ferreira (2017) define que em aplicações de reconhecimento de objetos, por exemplo, é comum as primeiras camadas de convolução detectarem bordas ou manchas que seriam as características mais básicas da imagem. As camadas de convolução mais profundas detectam outras características mais específicas.

Uma camada de convolução possui alguns atributos essenciais. O primeiro deles, é o tamanho do filtro. Nas principais CNNs os filtros geralmente são 1×1 ou 3×3, porém algumas redes podem possuir filtros maiores como 5×5 ou 7×7. O segundo parâmetro é o *stride* que define o tamanho do salto que será feito pelo filtro durante as operações de convolução. No processamento digital de imagens normalmente esse salto é de um *pixel*, porém, nas CNNs pode ser vantajoso dar saltos maiores.

O último parâmetro é o *padding*. O *padding* consistem em preencher os cantos da imagem com zeros, de forma a aumentar o tamanho da imagem sem alterar o seu conteúdo. Esse preenchimento pode ser feito por duas razões principais: a primeira delas é porque as operações de convolução reduzem o tamanho da imagem e a principal forma de evitar essa redução de tamanho é o uso de *padding*; a segunda razão também tem a ver com a redução de tamanho. Algumas vezes, é necessário reduzir o tamanho dos mapas em uma rede convolutiva, e neste sentido, as camadas de convolução (ou de *pooling*) atuam

fazendo o *downsample* - ou redução de amostragem. Mas, em alguns casos o tamanho da saída não será um valor inteiro, sendo necessária uma correção. Essa correção pode ser feita via corte - quando uma ou mais linhas ou colunas são cortadas durante a convolução - ou por preenchimento, quando uma ou mais linhas ou colunas de zeros são acrescentadas. O corte e o preenchimento são chamados respectivamente de *padding* válido e *padding* equivalente.

Sempre ligada a uma camada de convolução, é comum ter uma camada com função de ativação. Essas camadas geralmente usam as mesmas funções descritas na Seção 2.2.1.

As camadas de *pooling* são utilizadas para reduzir a dimensão de dados vindo das camadas de convolução, consequentemente reduzindo o custo computacional (FERREIRA, 2017). Na prática elas funcionam como as camadas de convolução, mas ao invés de fazer a operação tradicional de convolução, elas reduzem um conjunto de elementos a um único representante com base em uma função específica. As funções mais utilizadas são máximo e média (FERREIRA, 2017).

Por fim, as camadas completamente conexas são exatamente iguais às redes MLP e é comum utilizá-las no final da rede para conectar à saída. Seu acréscimo no final é importante pois faz a ligação de todos os filtros (FERREIRA, 2017). As arquiteturas de CNNs geralmente intercalam algumas camadas de convolução com uma camada de *pooling*. As arquiteturas mais antigas costumavam adicionar camadas completamente conexas ao final, porém, os modelos mais novos, como a *Residual Network* - Rede Neural Residual (ResNet) e a DenseNet já não usam mais esse conceito.

2.3.2 Regularização

Caelles et al. (2017) definem que um grande problema de trabalhar com modelos de *Deep Learning* é a necessidade de um grande volume de dados para que o modelo possa generalizar bem. Do contrário, pode acontecer *Overfitting*. Em *Deep Learning*, *Overfitting* é quando o modelo aprende a resolver o problema apenas no conjunto de dados de treinamento, isto é, gera uma solução pouco genérica.

Goodfellow, Bengio e Courville (2016) definem que uma das principais formas de evitar o *Overfitting* são as técnicas de regularização. Essas técnicas criam certas restrições dentro do modelo, e essas restrições dificultam a ocorrência do *Overfitting*.

Ioffe e Szegedy (2015) propuzeram uma técnica de regularização que se ajustou muito bem em redes neurais convolutivas e melhoraram os resultados da *Inception* (Szegedy et al., 2015) no desafio da ILSVRC. Essa Técnica foi o *Batch Normalization*, e ela consiste, basicamente em subtrair a média do lote de processamento da função de ativação, e dividir o resultado pelo desvio padrão. Desde que proposto, o *Batch Normalization*

se tornou presente nas arquiteturas mais profundas que surgiram depois, como ResNet (He et al., 2016) e DenseNet (Huang et al., 2017).

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os trabalhos na literatura que possuem relação com o trabalho proposto.

3.1 DenseNet

Como mencionado anteriormente, desde 2012 as tarefas de classificação de imagens são resolvidas majoritariamente por arquiteturas de *Deep Learning*. Em 2016, He et al. propuseram a ResNet que introduziam o conceito de *skip connection*. Uma *skip connection* é quando a camada C_n além de se conectar com a camada C_{n+1} ela se conecta com outra mais a frente - no caso da *ResNet*, a camada C_{n+3}. Sendo assim, a camada C_{n+3} recebe em suas entradas uma junção das saídas de C_n e C_{n+2}. Essa combinação de entradas na ResNet foi feita como uma soma de matrizes. A Figura 2 mostra um exemplo de *skip connection* na *ResNet*.

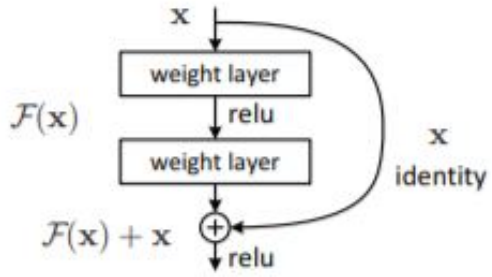
Seguindo a mesma ideia de *skip connection* proposta por He et al. (2016), Huang et al. (2017) propuseram a DenseNet. Nessa nova arquitetura houveram duas mudanças com relação à ResNet. A primeira mudança é que as saídas das camadas não seriam mais somadas, mas sim concatenadas, ou empilhadas. A segunda é que em um bloco residual as camadas se conectam com todas as suas sucessoras.

A Figura 3 mostra como se conectam as camadas na DenseNet. Supondo que temos um bloco denso B com um conjunto de 5 camadas $C = \{c_1, c_2, c_3, c_4, c_5\}$. Como explicado no parágrafo anterior, a saída das camadas serão utilizadas em todas as camadas subsequentes no mesmo bloco. Sendo assim, a entrada de c_3 será composta pelas saídas de c_1 e c_2 concatenadas, e assim sucessivamente. Além disso, como as saídas são concatenadas e não somadas, a tendência é que as entradas cresçam a medida que a rede se aprofunda. O conceito de bloco denso permite propagar múltiplos níveis de abstração ao longo de toda a rede, melhorando assim os resultados obtidos (Huang et al., 2017).

Sendo assim, se cada camada do bloco possui uma saída de x matrizes, o bloco possui n camadas e a entrada do bloco possui tamanho a, a saída possuirá o tamanho representado pela Equação 3.1. Vale mencionar que o valor de x se conserva para toda a rede.

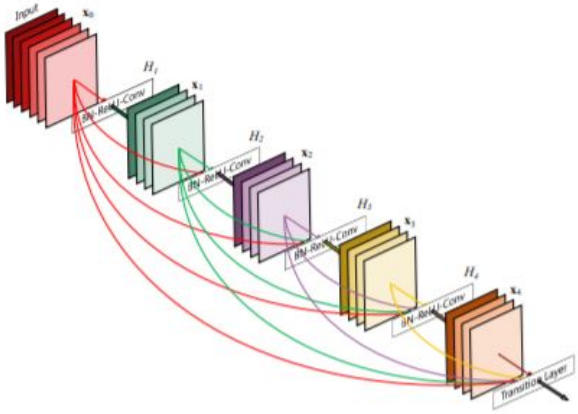
$$y = nx + a \tag{3.1}$$

Figura 2 – Exemplo de bloco residual



Fonte: He et al. (2016).

Figura 3 – Exemplo de bloco denso

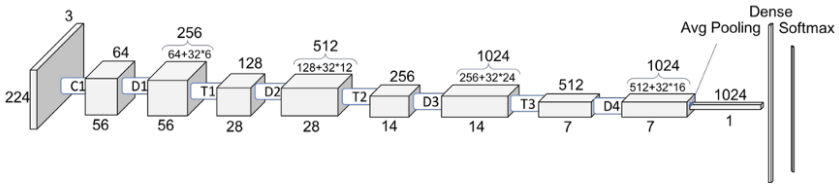


Fonte: Huang et al. (2017).

As DenseNets seguem um padrão arquitetural muito similar ao das ResNets começando a rede com uma camada de convolução 7×7 com *strides* de 2×2 que é sucedida por uma camada de *batch normalization* e outra de ReLU. Depois disso, têm uma camada de *pooling* pelo máximo e após essa camada, têm quatro blocos densos intercalados por blocos intermediários. Os blocos de convolução dentro dos blocos densos são compostos por uma camada de convolução 1×1 (que Huang et al. (2017) definem como camada de "gargalo") e uma camada de convolução 3×3. Vale mencionar que todas as camadas de convolução nos blocos densos são precedidas por uma camada de *batch normalization* e ReLU. Os blocos intermediários são compostos por uma camada de convolução 1×1 seguida de uma camada de *pooling* pela média (que também é precedida por *batch normalization* e ReLU). No final dos 4 blocos densos, a rede possui mais uma camada de *batch normalization* e ReLU seguida por um *pooling* pela média global e uma camada completamente conectada que faz a classificação. A Figura 4 mostra como é a arquitetura da rede.

O modelo proposto obteve resultados competitivos com os da literatura, ficando em segundo lugar no ILSVRC de 2017 e apresentando resultados competitivos com os da literatura na época. O artigo também foi premiado como a melhor publicação da *Conference on Computer Vision and Pattern Recognition* - Conferência de Visão Compu-

Figura 4 – Arquitetura DenseNet



Fonte: Huang et al. (2017).

tacional e Reconhecimento de Padrões (CVPR) do ano de 2017. O modelo obteve uma acurácia de 96% na Cifar-10 e 82% na Cifar-100.

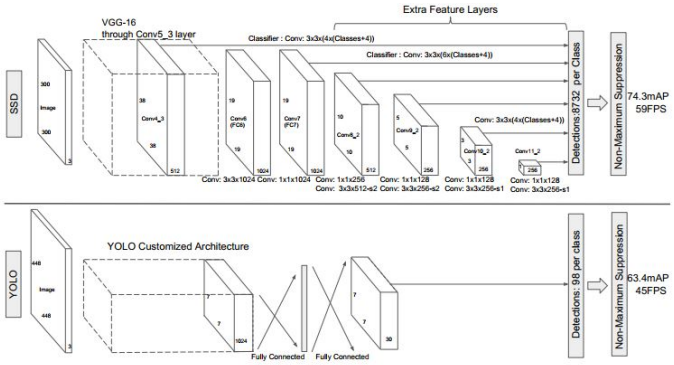
3.2 SSD: *Single-Shot Multibox Detector*

Liu et al. (2015) propuseram um método a base de redes neurais convolucionais para fazer a localização e detecção de objetos. A proposta deles melhorou significativamente os resultados apresentados no estado da arte. Isso se deve ao fato de que eles não só conseguiram propor um modelo que faz a localização e classificação de forma eficiente (chegando a 74,3% de mAP), como conseguiram obter esse resultado fazendo classificação e localização em tempo real, com uma velocidade de 59 FPS. Uma outra vantagem obtida por esse método é que ele consegue fazer a localização e classificação em imagens significativamente menores, uma vez que os quadros processados pelo YOLO têm dimensões 448×448 e os processados pelo R-CNN têm 1000×600, o SSD processa quadros de dimensões 300×300. Isso é uma vantagem, pois mesmo o algoritmo trabalhando com imagens menores, o resultado consegue competir com os outros métodos.

A abordagem consiste em utilizar a rede VGG16 (SIMONYAN; ZISSERMAN, 2014) como arquitetura base, substituir as camadas completamente conectadas fc6 e fc7 por camadas convolucionais, alterar o filtro pool5 de $2 \times 2 - s2$ para $3 \times 3 - s1$, e usaram o algoritmo *à trous*(HOLSCHNEIDER et al., 1990) para preencher os espaços vazios. Além disso, eles removeram todas as camadas de *dropout* e a última camada completamente conectada. Por fim, as camadas de convolução geram os resultados de mais de 8000 localizações e classificações, as quais são filtradas em um passo final de supressão de não-máximos, que elimina todos os resultados com confiança abaixo de $0,5$.

A Figura 5 mostra as diferenças entre as arquiteturas YOLO e SSD. Enquanto Redmon et al. (2015) usaram uma camada completamente conectada intermediária para fazer a localização dos objetos, Liu et al. (2015) usaram camadas de convolução sobre mapas de múltiplos tamanhos. Além disso, o SSD trabalha com *bounding-boxes* de tamanhos padrões $\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. Os filtros de convolução adicionais, os tamanhos padrões de *bounding-boxes* e o uso de *data-augmentation* foram cruciais na obtenção dos bons resultados.

Figura 5 – Comparação entre YOLO e SSD



Fonte: Liu et al. (2015).

3.3 DSSD: *Deconvolution Single-Shot Detector*

Fu et al. (2017) propuseram uma extensão do SSD. Depois dos resultados obtidos pelo SSD ao fazer localização e classificação de objetos com uma mAP de $79,5\%$, eles propuseram uma abordagem alternativa, usando camadas de deconvolução ao final da rede. As camadas de deconvolução tem entre seus resultados o aumento de resolução do mapa de entrada. A abordagem visou explorar esse efeito com o intuito de aumentar a mAP da classificação e localização de objetos, e, com isso, atingir uma mAP de $81,5\%$. Embora essa abordagem tenha uma mAP maior do que a obtida por Liu et al. (2015), ela não é rápida o bastante pra fazer localização e classificação em tempo real.

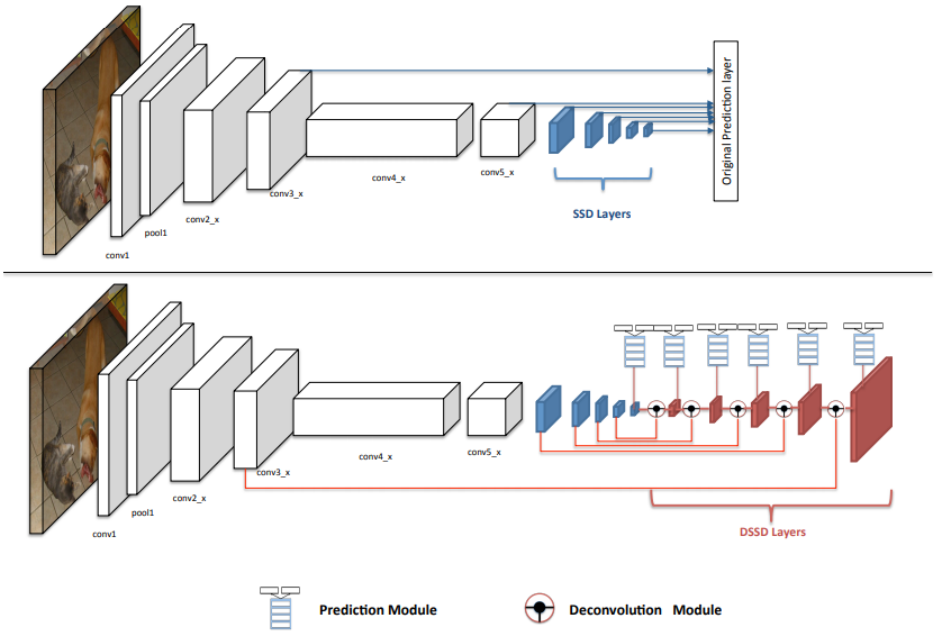
A Figura 6 mostra as arquiteturas da SSD e DSSD mostrando as suas diferenças. É possível visualizar que a rede DSSD utiliza as saídas da SSD como entrada aos módulos de deconvolução.

Nesse trabalho, foram propostas duas alterações principais no modelo SSD. A primeira delas foi a utilização da rede neural ResNet-101 (He et al., 2016). Fu et al. (2017) perceberam, porém, que apenas substituir a VGG16 pela ResNet não produziria resultados melhores. Sendo assim, após adicionar as camadas de convolução do SSD para fazer a detecção em múltiplos níveis, ele adicionou um módulo com mais algumas camadas de convolução a fim de melhorar os resultados. Os módulos propostos estão na Figura 7.

Fazendo os testes, Fu et al. (2017) perceberam que os melhores resultados foram obtidos usando a terceira opção que consistem um módulo residual com uma convolução na conexão alternativa e a junção das duas entradas sendo feita por meio de uma multiplicação elemento por elemento. Neste ponto, as mudanças ainda se mostravam pouco efetivas, já que a nova SSD321 apresentava 77,1% de mAP contra 77,5% de mAP da SSD300, ao passo que a nova SSD513 apresentava 80,6% contra 79,5% da SSD512.

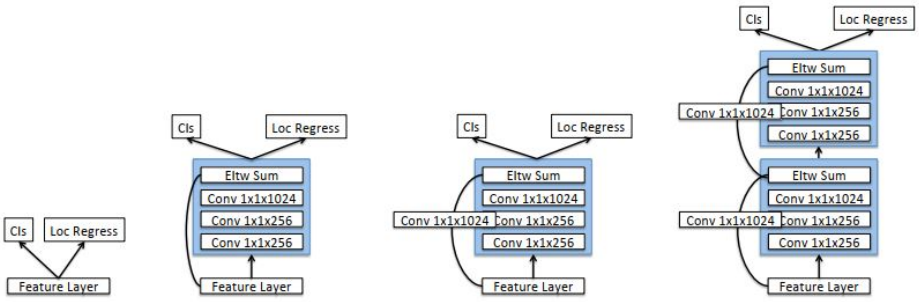
Tendo isso em vista, após a implementação dos módulos SSD na ResNet101, Fu et al. (2017) decidiram adicionar módulos usando camadas de deconvolução com o intuito

Figura 6 – Comparação entre SSD e DSSD



Fonte: Fu et al. (2017).

Figura 7 – Módulos de predição SSD com ResNet

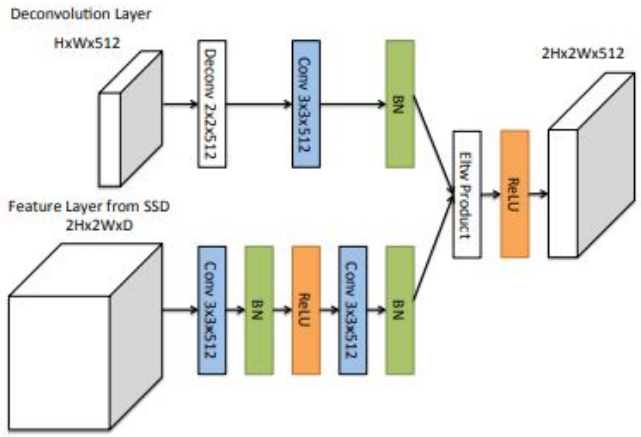


Fonte: Fu et al. (2017).

de fazer um *upsample* nas saídas da SSD para obter mais dados. Depois de adicionar esses módulos, eles são combinados com o módulo SSD anterior para gerar a nova saída. A Figura 8 mostra como é feita essa combinação.

Como mencionado anteriormente, embora o DSSD tenha melhorado a mAP do SSD, ele não é aplicável para fazer a localização e classificação em tempo real. Com uma mAP de $81,5\%$ ele consegue processar apenas $6,6$ FPS. Isso se deve ao uso das deconvoluções, da ResNet 101 - que possui mais camadas, e, portanto toma mais tempo de processamento - e também ao aumento no número de *bounding boxes* geradas(43688 vs. 17080), fazendo assim com que a supressão de não-máximos leve mais tempo.

Figura 8 – Módulos de deconvolução



Fonte: Fu et al. (2017).

3.4 PASCAL VOC

Um grande desafio em treinar modelos de *Deep Learning* é a necessidade de um grande volume de dados classificados manualmente. Com essa necessidade, diversos trabalhos foram desenvolvidos com o objetivo de apresentar uma base de dados anotados manualmente para um conjunto de problemas específicos. Nos trabalhos de Localização e classificação, uma das bases mais utilizadas é a *PASCAL Visual Object Classes* (PASCAL VOC).

O desafio PASCAL VOC foi realizado anualmente entre 2004 e 2012, e a cada ano novas imagens e novos desafios foram propostos na área de visão computacional. Os principais desafios envolviam Localização e Classificação de Objetos, Segmentação Semântica, *layout* humano (um desafio que consiste em detectar partes do corpo como cabeça, mãos e pés) e classificação de ações (EVERINGHAM et al., 2015).

O desafio de Localização e Classificação de objetos desde 2007 possui 20 classes distintas de objetos e a cada ano eles criaram uma nova base com mais imagens extraídas da internet. A Tabela 1 mostra as classes, o número de imagens e o número de objetos por classe nos conjuntos de treino, validação e teste.

Tabela 1 – Classes de imagens do PASCAL VOC 2007

	Treino		Validação		Treino/Validação		Teste	
	imagens	Objetos	Imagens	Objetos	Imagens	Objetos	Imagens	Objetos
Avião	112	151	126	155	238	306	204	285
Bicicleta	116	176	127	177	243	353	239	282
Pássaro	180	243	150	243	330	486	282	459
Barco	81	140	100	150	181	290	172	263
Garrafa	139	253	105	252	244	505	212	469
Ônibus	97	115	89	114	186	229	174	213
Carro	376	625	337	625	713	1250	721	1201
Gato	163	186	174	190	337	376	322	358
Cadeira	224	400	221	398	445	798	417	756
Vaca	69	136	72	123	141	259	127	244
Mesa de Jantar	97	103	103	112	200	215	190	206
Cão	203	253	218	257	421	510	418	489
Cavalo	139	182	148	180	287	362	274	348
Motocicleta	120	167	125	172	245	339	222	325
Pessoa	1025	2358	983	2332	2008	4690	2007	4528
Vaso de planta	133	248	112	266	245	514	224	480
Ovelha	48	130	48	127	96	257	97	242
Sofá	111	124	118	124	229	248	223	239
Trem	127	145	134	152	261	297	259	282
Monitor/TV	128	166	128	158	256	324	299	308
Total	2501	6301	2510	6307	5011	12608	4952	12032

Fonte: Adaptado de Everingham et al. (2010).

Analisando a tabela é possível perceber que a base é desbalanceada. A classe pessoa, por exemplo, está presente em mais de 1000 imagens e tem mais de 2000 instâncias apenas no conjunto de treinamento. Enquanto isso, a classe ovelha aparece apenas 130 vezes em 48 imagens no conjunto de treinamento. Esse desbalanceamento pode ser um desafio durante o treinamento caso o modelo não seja robusto. A Figura 9 mostra exemplos das classes presentes na PASCAL VOC 2007.

3.4.1 PASCAL VOC 2012

Em 2012 foi lançado uma outra versão da PASCAL VOC. Essa edição da competição tinha uma base de dados maior e mais robusta, embora ainda mantivesse as mesmas 20 classes da base de 2007. A Tabela 2 mostra a distribuição de imagens e objetos na base PASCAL VOC 2012. Nela será possível perceber o mesmo desbalanceamento encontrado na base de 2007. Porém, uma vantagem é que ela tem mais imagens, se comparada à base de 2007. Além disso, como as bases são completamente distintas, podem ser usadas em conjunto para o treinamento e a validação dos modelos.

Tabela 2 – Classes de imagens do PASCAL VOC 2012

	Treino		Validação		Treino/Validação	
	imagens	Objetos	Imagens	Objetos	Imagens	Objetos
Avião	327	432	343	433	670	865
Bicicleta	268	353	284	358	552	711
Pássaro	395	560	370	559	765	1119
Barco	260	426	248	424	508	850
Garrafa	365	629	341	630	706	1259
Ônibus	213	292	208	301	421	593
Carro	590	1013	571	1004	1161	2017
Gato	539	605	541	612	1080	1217
Cadeira	566	1178	553	1176	1119	2354
Vaca	151	290	152	298	303	588
Mesa de Jantar	269	304	269	305	538	609
Cão	632	756	654	759	1286	1515
Cavalo	237	350	245	360	482	710
Motocicleta	265	357	261	356	526	713
Pessoa	1994	4194	2093	4372	4087	8566
Vaso de planta	269	484	258	489	527	973
Ovelha	171	400	154	413	325	813
Sofá	257	281	250	285	507	566
Trem	273	313	271	315	544	628
Monitor/TV	290	392	285	392	575	784
Total	5717	13609	5823	13841	11540	27450

Fonte: Adaptado de Everingham e Winn (2011).

3.4.2 Métricas

A principal métrica utilizada para avaliar trabalhos de Localização e Classificação é a mAP. Para deduzir o mAP, é necessário definir os conceitos de Precisão e Revocação. Então seja TP os verdadeiros positivos encontrados por um modelo, FN os falsos negativos e FP os falsos positivos (os verdadeiros negativos não entram no cálculo pois os modelos não são treinados com base no *ground-truth* do *background*). A Precisão P pode ser definida pela Equação 3.2.

$$P = \frac{TP}{TP + FP} \tag{3.2}$$

Ao passo que a revocação R pode ser definida pela Equação 3.3.

$$R = \frac{TP}{TP + FN} \tag{3.3}$$

Uma vez que se tenha os valores de precisão e revocação, é traçada a curva $P \times R$.

Essa curva se comporta com R aumentando a medida que aumentam os falsos negativos e com P oscilando e fazendo "zigue-zague". O valor teórico da Precisão média é determinado pela área sobre a curva $P \times R$ como está descrito na Equação 3.4.

$$AP = \int_0^1 P(R)dR \tag{3.4}$$

Porém, como mencionado anteriormente, a curva $P \times R$ se comporta de forma não-linear fazendo "zigue-zague". Esse comportamento dificulta um cálculo preciso da área sobre a curva. Para contornar essa dificuldade, Salton e McGill (1986) propuseram que o valor da precisão média fosse interpolado de acordo com a Equação 3.5

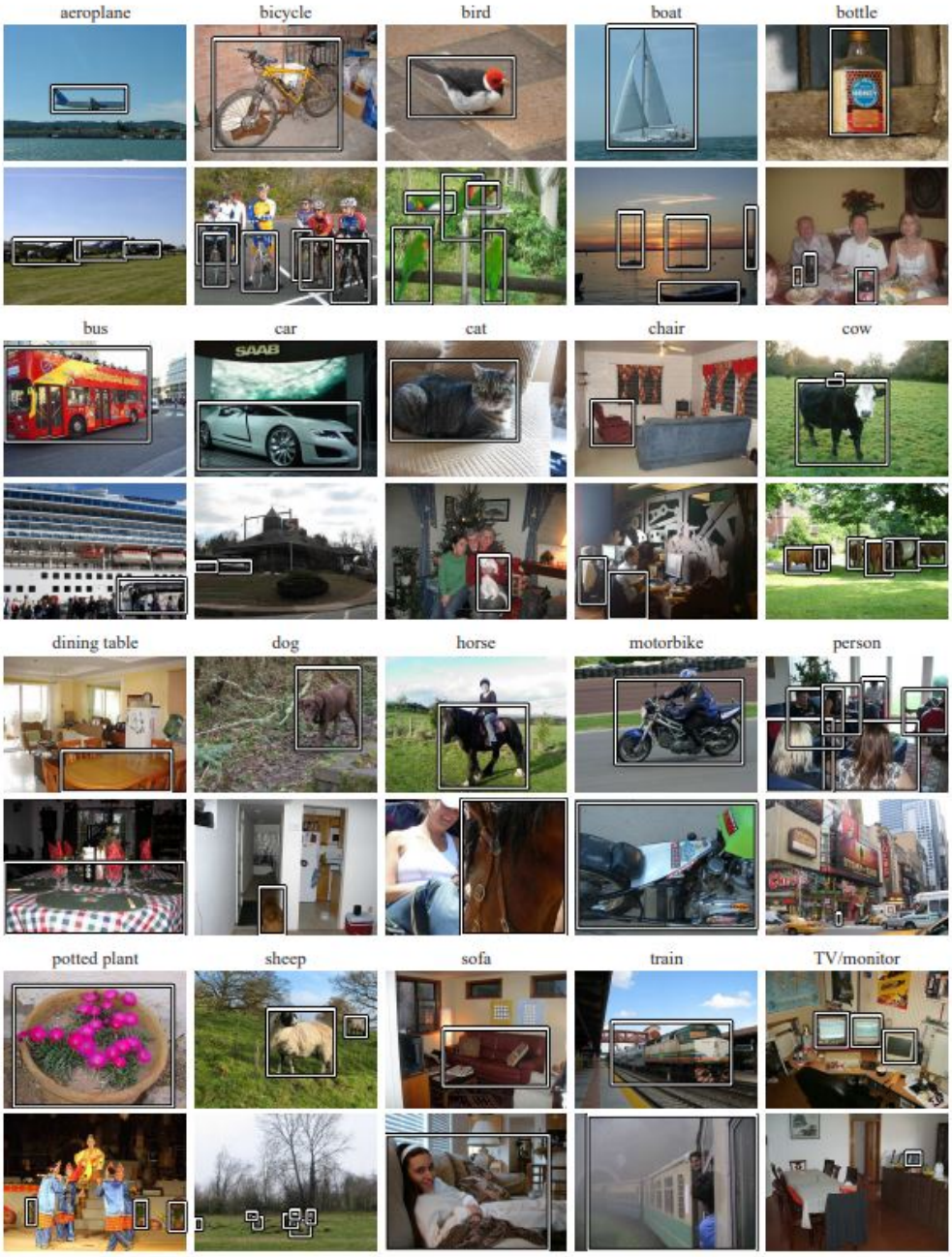
$$AP = \frac{1}{11} \sum_{R_i \in \{0.0,0.1,..1.0\}} max(P(R_i)) \tag{3.5}$$

Everingham et al. (2015) porém começaram a adotar uma forma diferente de calcular a integral a partir de 2010. Ao invés de fazer a interpolação como na Equação 3.5, eles passaram a calcular a integral pela regra dos trapézios, definida pela Equação 3.6.

$$AP = \frac{0.1}{2}[P(R_0) + P(R_1) + 2. \sum_{R_i \in \{0.1,0.2,..,0.9\}} P(R_i)] \tag{3.6}$$

Sendo assim, o mAP pode ser definido como a média das precisões médias por classe.

Figura 9 – Exemplos de imagens da PASCAL VOC 2007



Fonte: Everingham et al. (2010).

4 PROPOSTA TÉCNICA

Neste capítulo serão descritas a metodologia, o ambiente de testes e as tecnologias usadas no desenvolvimento do trabalho.

4.1 Metodologia

Nesta seção será descrita a metodologia necessária para o desenvolvimento do trabalho.

4.1.1 Levantamento Bibliográfica

Nessa etapa foi feita um levantamento bibliográfico. O objetivo é entender como funciona na teoria e na prática os algoritmos de localização e classificação de objetos. Além disso, o levantamento também serviu para definir a base de dados a ser utilizada no trabalho, a arquitetura de redes neurais convolutivas a ser implementada, a abordagem utilizada para fazer a localização e classificação de objetos e a métrica utilizada para avaliar o trabalho.

A arquitetura base selecionada foi a DenseNet121, as bases de imagens selecionadas foram as versões de 2007 e 2012 da PASCAL VOC. A abordagem selecionada para fazer a localização e classificação envolve elementos da SSD (LIU et al., 2015) e da DSSD (FU et al., 2017). Por fim, a métrica utilizada é a mAP conforme definida na Seção 3.4.2.

4.1.2 Implementação e treinamento

Nesta fase foi feita a implementação da DenseNet121 com as modificações propostas por Liu et al. (2015) e Fu et al. (2017) para fazer a localização e classificação dos objetos. Feita a implementação da DenseNet e das modificações, o modelo foi treinado usando a base de dados PASCAL VOC nas versões 2007 e 2012. Mais detalhes das modificações feitas na arquitetura e do treinamento podem ser encontradas na Seção 4.3.

4.1.3 Testes e Avaliação

Nesta fase foram feitos os testes com o modelo proposto. Eles foram feitos usando o subconjunto de testes da PASCAL VOC 2012. Após os testes, foi feita a avaliação dos

resultados do modelo com a métrica proposta (mAP). Em cima dos resultados obtidos foi feita uma comparação com os principais trabalhos relacionados na literatura (LIU et al., 2015; FU et al., 2017) com o intuito de apontar as principais falhas e os principais ganhos do modelo proposto.

Além disso, foram feitas algumas demonstrações com o intuito de gerar imagens para se fazer uma avaliação qualitativa dos resultados obtidos pelo modelo. Por fim, foi escrita uma monografia contendo todo o conteúdo dos estudos realizados, das implementações feitas e dos resultados obtidos. O Capítulo 5 apresenta de forma mais detalhada os resultados obtidos nos experimentos feitos.

4.2 Ambiente de teste

Todo o código foi implementado usando as seguintes tecnologias:

- Python 3.5.2;
- TensorFlow 1.8.0;
- Keras 2.2.4;
- Jupyter 2.1.2;
- Notebook 6.0.3.

Os experimentos foram realizados no computador com as seguintes configurações:

- **Sistema Operacional:** Ubuntu 18.04 LTS;
- **CPU:** Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz (6 núcleos, duas *threads* por núcleo);
- **Memória RAM:** 32GB;
- **GPU:** NVidia GTX 1080.

4.3 Implementação e treinamento

Nesta seção será descrita de forma mais detalhada a implementação das modificações na arquitetura da rede neural, e o treinamento.

4.3.1 Modificação da Arquitetura

A implementação consiste em adaptar a DenseNet 121 para realizar as tarefas de localização e classificação. Nesse estágio foi feita uma pequena alteração com relação aos modelos propostos por Liu et al. (2015) e por Fu et al. (2017). Enquanto que Liu et al. (2015) usou imagens de tamanho 300×300 e Fu et al. (2017) usou imagens de tamanho 321×321, foi usado nesse trabalho imagens de 311×311.

O motivo é que, embora a DenseNet possua as mesmas características de *downsample* (por meio de convolução ou *pooling*) que a ResNet, a implementação da SSD por Fu et al. (2017) usa *padding* válido para todas as camadas de *downsample*. Isso faz com que, caso o tamanho de saída da camada de *downsample* não seja inteiro, uma parte seja cortada, podendo causar perda de informação. Sendo assim, foi feita a opção por uma imagem com tamanho menor e o uso de *padding* equivalente, de forma a evitar o corte.

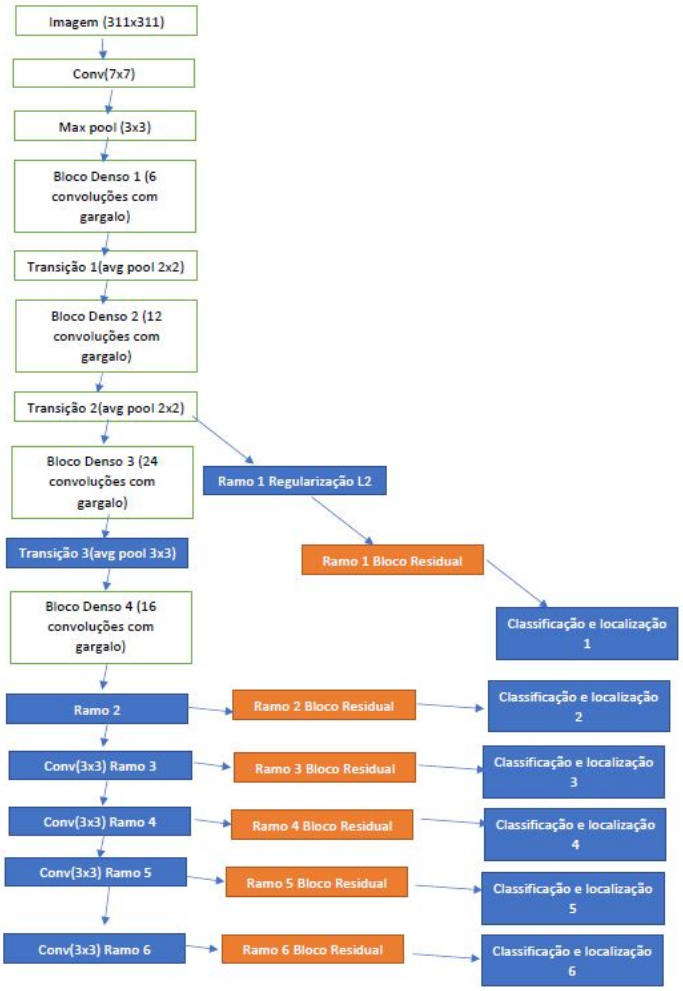
Além dessas modificações, é necessário mudar algumas coisas na arquitetura da rede DenseNet121. A primeira delas é que a SSD utiliza ramificações de diferentes níveis para fazer as predições de forma mais precisa. Sendo assim, a primeira alteração é que no segundo bloco de transição é criada uma ramificação fazendo regularização L2. Além disso, a camada de *pooling* do último bloco de transição é substituída por um bloco 3×3 com strides igual a 1. No final da rede as camadas de *pooling* global e softmax são removidas e essa saída é a segunda ramificação da SSD. Depois disso, são adicionados mais dois blocos com camadas de convolução, uma 1×1 seguida de um 3×3 com *stride* 2. Por fim, mais dois blocos, com as mesmas camadas, porém, as camadas 3×3 com *stride* igual a 1. A Figura 10 mostra a nova arquitetura da rede. Os blocos brancos representam as partes da implementação original da DenseNet121, os blocos azuis representam as modificações propostas por Liu et al. (2015) e os blocos laranjados representam as modificações propostas por Fu et al. (2017).

4.3.2 Treinamento

A rede foi inicializada com os pesos da DenseNet121 treinada na ImageNet, e as camadas adicionais foram inicializadas com o método *He normal* proposto por He et al. (2015). O algoritmo de treinamento selecionado foi o Adam, proposto por Kingma e Ba (2014) com os seguintes hiperparâmetros:

- $\alpha = 5 \times 10^{-5}$;
- $\beta_1 = 0,9$;
- $\beta_2 = 0,999$;

Figura 10 – Arquitetura com modificações propostas para a DenseNet



Fonte: Elaborado pelo autor.

- $\epsilon = 1 \times 10^{-8}$.

Além disso, o treinamento ocorreu em um total de 400 épocas, cada época com 1000 iterações e cada iteração com um lote de processamento de 12 imagens. Após 200 épocas de treinamento, a taxa de aprendizado caiu para 5×10^{-6} e após mais 100 épocas, caiu para 5×10^{-7}. As bases de dados usadas para os testes foram a PASCAL VOC 2007 e 2012. Para treinamento foram usadas as imagens do conjunto de treino e validação de ambas as bases, e para validação do treinamento foi utilizada a base de testes de 2007. Os testes foram feitos na base de testes de 2012.

5 RESULTADOS

Neste capítulo serão feitas as avaliações qualitativas e quantitativas do modelo proposto, além de uma comparação com o Estado da Arte.

5.1 Avaliação Qualitativa

Depois do treinamento, foram feitos alguns experimentos com imagens aleatórias do conjunto separado para validação para avaliar os resultados obtidos pelo método. Embora o método consiga fazer a detecção e a classificação dos objetos, foram percebidas algumas dificuldades.

Na Figura 11 o modelo detectou a pessoa sentada na cadeira e a cadeira no fundo, porém não detectou o cão deitado no chão. Isso pode refletir um pouco de dificuldade em perceber objetos menores ou que não estejam tão centralizados. Além disso, Embora a pessoa sentada esteja bem nítida na imagem, o modelo não conseguiu ter 100% de confiança que era uma pessoa, o que mostra um certo nível de incerteza na predição.

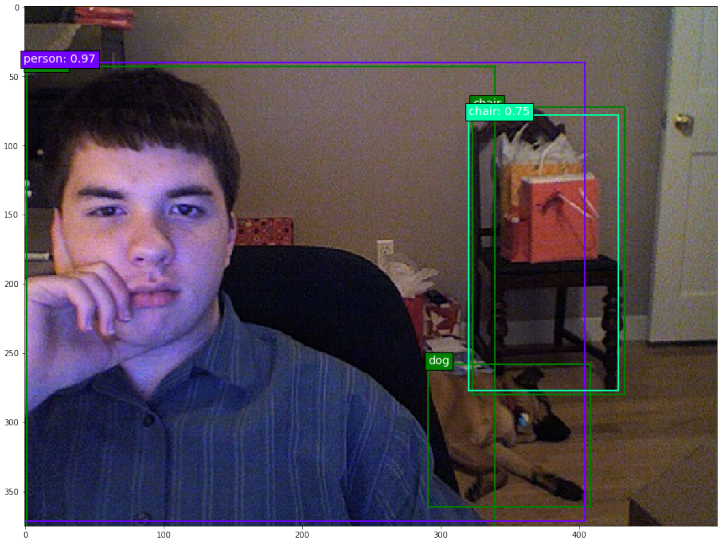
Na Figura 12 percebemos mais uma vez o problema da confiança baixa. Além disso, a *bounding box* do cão de trás esta pegando apenas a cabeça, sendo que outras partes do corpo dele estão expostas (ainda que sobrepostas parcialmente).

Na Figura 13, mostra mais uma vez uma nítida dificuldade do modelo em perceber objetos mais distantes. A imagem possui 9 objetos delimitados no *ground-truth* e o modelo conseguiu detectar apenas 2. Ele não consegue perceber nem a mesa de jantar e nem as pessoas mais ao fundo. Além disso, as duas pessoas que estão em pé no canto direito da Figura foram percebidas como apenas uma pelo modelo.

A Figura 14 é bem simples e o único objeto presente está bem nítido, então o modelo consegue percebê-lo com a confiança de 100%.

Já a Figura 15 mostra um fenômeno diferente: além de detectar o único objeto presente no *ground-truth* com a confiança alta, ele também detecta uma outra instância do mesmo objeto ao fundo. Isso significa que embora o modelo tenha um pouco de dificuldade com objetos que não estão em destaque, essa dificuldade não é tão crítica e existe uma certa robustês.

Figura 11 – Resultado dos testes 1



Fonte: Elaborado pelo autor.

5.2 Avaliação Quantitativa

Depois dos experimentos, foram feitos os testes para avaliar de forma numérica os resultados do modelo e comparar com o estado da arte. O resultado final apontou um mAP de 72,6%.

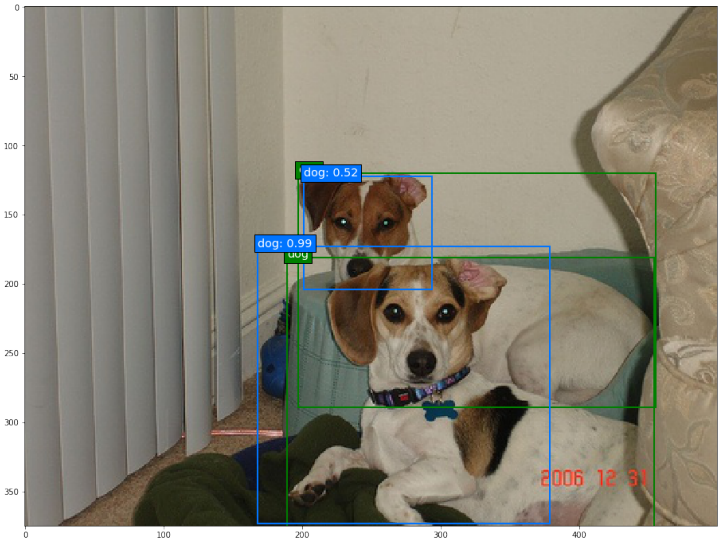
Com base na Tabela 3 é possível ver que os resultados estão um pouco abaixo dos obtidos por Liu et al. (2015) e Fu et al. (2017). Apesar disso, pode-se observar que o modelo se saiu muito bem em algumas classes como Avião, Carro, Motocicleta, Cavalo e bicicleta, chegando a ter resultados melhores que os outros modelos em algumas delas.

A principal dificuldade está nos objetos mais difíceis, onde o modelo não conseguiu se ajustar bem, e acabou apresentando resultados muito inferiores aos obtidos pelos outros métodos. Entre essas classes, é possível destacar o vaso de planta, a cadeira, e a Garrafa. Algumas dessas classes o modelo perde por uma discrepância de mais de 10 pontos percentuais.

A diferença nos resultados se deve a alguns fatores. Entre eles é possível citar o tamanho do lote de treinamento. Enquanto o modelo proposto foi treinado com 12 imagens por lote, os modelos de Liu et al. (2015) e Fu et al. (2017) foram treinados usando um lote de 32 imagens.

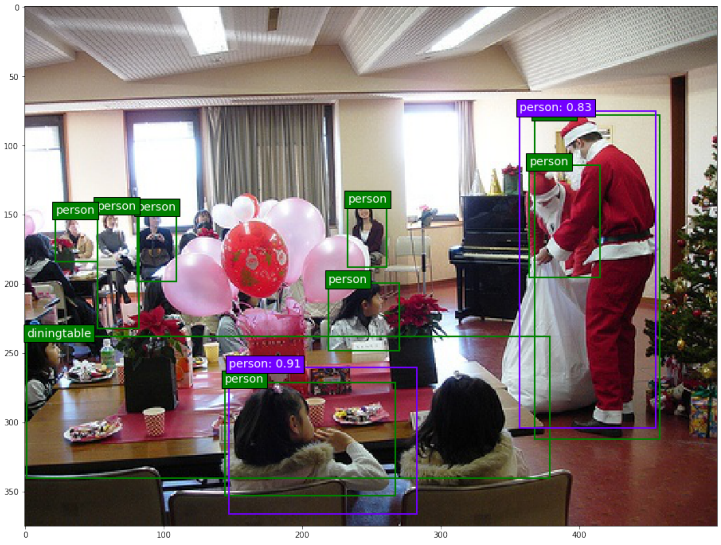
Além disso, devido ao número menor no lote de treinamento, foi necessário aumentar o número de épocas e iterações para garantir a convergência. O aumento de iterações com lotes menores pode causar um *Overfitting* que é quando o modelo aprende os dados específicos ao invés de aprender a generalizar com base nos dados.

Figura 12 – Resultado dos testes 2



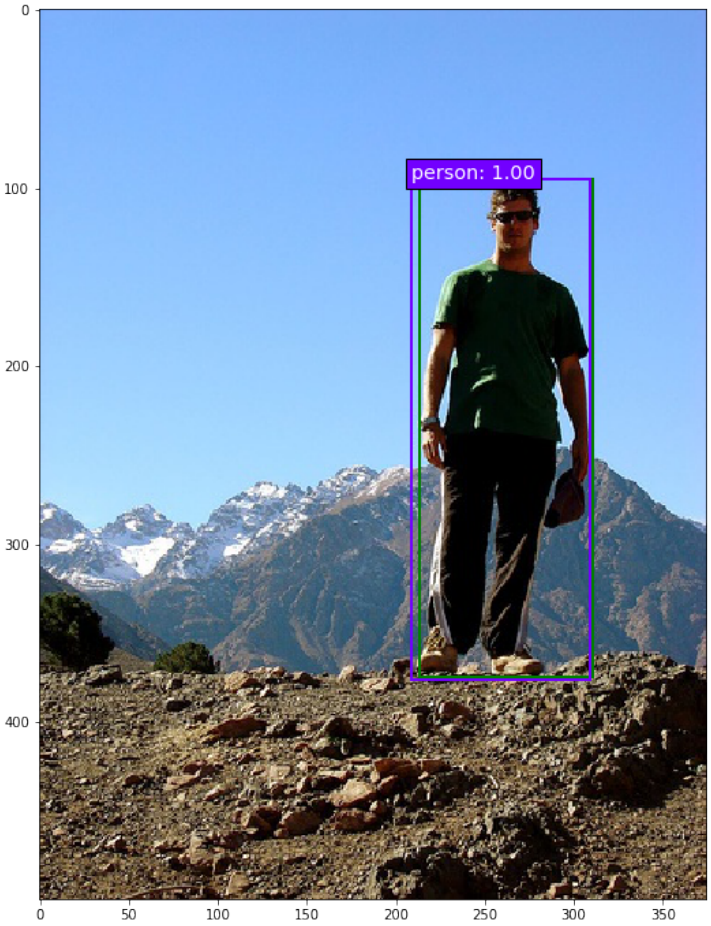
Fonte: Elaborado pelo autor.

Figura 13 – Resultado dos testes 3



Fonte: Elaborado pelo autor.

Figura 14 – Resultado dos testes 4



Fonte: Elaborado pelo autor.

Figura 15 – Resultado dos testes 5

Fonte: Elaborado pelo autor.

Tabela 3 – Comparação SSD300, SSD321 e SSD311

Objeto	SSD311(nosso)	SSD300	SSD321
Avião	80,8%	79,5%	76,3%
Bicicleta	82,3%	83,9%	84,6%
Pássaro	66,3%	76%	79,3%
Barco	61,3%	69,6%	64,6%
Garrafa	31,2%	57,5%	47,2%
Ônibus	82,8%	87%	85,4%
Carro	86%	85,7%	84%
Gato	85,8%	88,1%	88,8%
Cadeira	52,9%	60,3%	60,1%
Vaca	75%	81,5%	82,6%
Mesa de Jantar	72,4%	77%	76,9%
Cão	77,4%	86,1%	86,7%
Cavalo	86,2%	87,5%	87,2%
Motocicleta	86,3%	84%	85,4%
Pessoa	77,4%	79,4%	79,1%
Vaso de planta	42,6%	52,3%	50,8%
Ovelha	69,9%	77,9%	77,2%
Sofá	77,2%	79,5%	82,6%
Trem	84,3%	87,6%	87,3%
Monitor/TV	73%	76,8%	76,6%
Total	72,6%	77,5%	77,1%

Fonte: Elaborado pelo autor.

6 CONCLUSÃO

O uso de redes neurais convolutivas tem se tornado cada vez mais comum nos problemas relacionados a imagens, principalmente devido aos resultados alcançados. Isso faz que cada vez mais modelos e técnicas avançadas sejam desenvolvidas com o objetivo de melhorar os resultados.

Uma dificuldade, porém com os modelos de *Deep Learning* é a necessidade de um grande volume de dados para fazer o treinamento de forma adequada. Além disso, criar uma base de dados para classificação e regressão é custoso, pois todos os dados devem estar anotados manualmente. Além de precisar de um grande volume de dados, também é necessário que a base esteja bem balanceada ou os resultados podem ser prejudicados.

Já existem estratégias de normalização e regularização que ajudam o modelo a não ser tão penalizado pelo desbalanceamento da base. Além disso, uma estratégia bem simples e bem eficaz para ajudar a contornar o problema da falta de dados, é o *data augmentation*.

Os resultados apresentados, mostram que as redes neurais densas podem ser usadas para fazer as tarefas de Localização e Classificação de objetos, desde que o modelo seja bem ajustado e bem treinado. Os resultados obtidos foram ligeiramente inferiores aos da literatura, mas isso de deve, principalmente ao lote de imagens usado no treinamento do nosso modelo que foi bem menor do que o usado nos métodos comparados.

Apesar disso, é possível observar em algumas classes um resultado superior aos obtidos por Liu et al. (2015) e por Fu et al. (2017), mostrando assim, que o uso da DenseNet é bem promissor e que pode ser mais explorado no futuro.

6.1 Trabalhos Futuros

Como trabalhos futuros, espera-se fazer um teste com o modelo desenvolvido treinando-o com mais imagens por iteração. Além disso, espera-se também fazer uma implementação usando as camadas de deconvolução, ou outras formas mais avançadas de melhorar a precisão da classificação. Também propõe-se o uso de imagens com resoluções maiores e a avaliação de quantos quadros o modelo consegue processar por segundo.

REFERÊNCIAS

APTÉ, C.; WEISS, S. Data mining with decision trees and decision rules. Future Generation Computer Systems, v. 13, n. 2, p. 197 – 210, 1997. ISSN 0167-739X. Tradução nossa. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X97000216>>.

ARNOLD, L. et al. An introduction to deep learning. In: European Symposium on Artificial Neural Networks (ESANN). [S.l.: s.n.], 2011. Tradução nossa.

Caelles, S. et al. One-shot video object segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2017. p. 5320–5329. ISSN 1063-6919. Tradução nossa.

Deng, J. et al. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2009. p. 248–255. ISSN 1063-6919. Tradução nossa.

DENG, L.; YU, D. Deep learning: Methods and applications. Foundations and Trends® in Signal Processing, v. 7, n. 3–4, p. 197–387, 2014. Tradução nossa.

DO, T.-N.; POULET, F. Parallel multiclass logistic regression for classifying large scale image datasets. In: THI, H. A. L.; NGUYEN, N. T.; DO, T. V. (Ed.). Advanced Computational Methods for Knowledge Engineering. Cham: Springer International Publishing, 2015. p. 255–266. ISBN 978-3-319-17996-4.

DOSUALDO, D. G. Investigação de regressão no processo de mineração de dados. Dissertação de Mestrado — Universidade de São Paulo, Maio 2003.

EVERINGHAM, M. et al. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, v. 111, n. 1, p. 98–136, jan 2015. Tradução nossa.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. International Journal of Computer Vision, v. 88, p. 303–338, 06 2010.

EVERINGHAM, M.; WINN, J. The pascal visual object classes challenge 2012 (voc2012) development kit. Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep, v. 8, 2011.

FERREIRA, A. dos S. Redes Neurais Convolucionais Profundas na Detecção de Plantas Daninhas em Lavoura de Soja. Dissertação (Dissertação (Mestrado)) — Universidade Federal do Mato Grosso do Sul, Pos-Graduação em Ciências da Computação, 2017.

FU, C. et al. DSSD : Deconvolutional single shot detector. CoRR, abs/1701.06659, 2017. Tradução nossa. Disponível em: <<http://arxiv.org/abs/1701.06659>>.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, v. 36, n. 4, p. 193–202, Apr 1980. ISSN 1432-0770. Tradução nossa. Disponível em: <<https://doi.org/10.1007/BF00344251>>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). [S.l.: s.n.], 2015.

He, K. et al. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2016. p. 770–778. ISSN 1063-6919. Tradução nossa.

HOLSCHNEIDER, M. et al. A real-time algorithm for signal analysis with the help of the wavelet transform. In: COMBES, J.-M.; GROSSMANN, A.; TCHAMITCHIAN, P. (Ed.). Wavelets. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990. p. 286–297. ISBN 978-3-642-75988-8.

Huang, G. et al. Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2017. p. 2261–2269. ISSN 1063-6919. Tradução nossa.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: BACH, F.; BLEI, D. (Ed.). Proceedings of the 32nd International Conference on Machine Learning. Lille, France: PMLR, 2015. (Proceedings of Machine Learning Research, v. 37), p. 448–456. Disponível em: <<http://proceedings.mlr.press/v37/ioffe15.html>>.

JOST, I. Aplicação de *Deep Learning* em Dados Refinados para Mineração de Opiniões. Dissertação (Mestrado) — Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo., 2015.

KINGMA, D.; BA, J. Adam: A method for stochastic optimization. International Conference on Learning Representations, 12 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105. Tradução nossa. Disponível em: <<http://dl.acm.org/citation.cfm?id=2999134.2999257>>.

Lan, W. et al. Pedestrian detection based on yolo network model. In: 2018 IEEE International Conference on Mechatronics and Automation (ICMA). [S.l.: s.n.], 2018. p. 1547–1551. ISSN 2152-744X. Tradução nossa.

Lecun, Y. et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE, v. 86, n. 11, p. 2278–2324, Nov 1998. ISSN 0018-9219. Tradução nossa.

LIN, T. et al. Microsoft COCO: common objects in context. CoRR, abs/1405.0312, 2014. Tradução nossa. Disponível em: <<http://arxiv.org/abs/1405.0312>>.

LIU, W. et al. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015. Tradução nossa. Disponível em: <<http://arxiv.org/abs/1512.02325>>.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. CoRR, abs/1411.4038, 2014. Tradução nossa. Disponível em: <<http://arxiv.org/abs/1411.4038>>.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. Sistemas Inteligentes-Fundamentos e Aplicações, v. 1, n. 1, 2003.

Noh, H.; Hong, S.; Han, B. Learning deconvolution network for semantic segmentation. In: 2015 IEEE International Conference on Computer Vision (ICCV). [S.l.: s.n.], 2015. p. 1520–1528. ISSN 2380-7504. Tradução nossa.

PEREIRA, J. C.; GARSON, S.; ARAÚJO, E. G. Construção de um modelo para o preço de venda de casas residenciais na cidade de sorocaba-sp. Revista GEPROS, n. 4, p. 153, 2012.

REDMON, J. et al. You only look once: Unified, real-time object detection. CoRR, abs/1506.02640, 2015. Tradução nossa. Disponível em: <<http://arxiv.org/abs/1506.02640>>.

Ren, P.; Fang, W.; Djahel, S. A novel yolo-based real-time people counting approach. In: 2017 International Smart Cities Conference (ISC2). [S.l.: s.n.], 2017. p. 1–2. Tradução nossa.

Ren, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 39, n. 6, p. 1137–1149, June 2017. ISSN 0162-8828.

SALTON, G.; MCGILL, M. J. Introduction to Modern Information Retrieval. USA: McGraw-Hill, Inc., 1986. ISBN 0070544840.

SANTOS, A. de M. Tese de Doutorado, Investigando a combinação de técnicas de aprendizado semissupervisionado e classificação hierárquica multirrótulo. [S.l.]: Universidade Federal do Rio Grande do Norte, 2012.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. Tradução nossa.

Szegedy, C. et al. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S.l.: s.n.], 2015. p. 1–9. ISSN 1063-6919. Tradução nossa.

VOIGTLAENDER, P.; LEIBE, B. Online adaptation of convolutional neural networks for video object segmentation. CoRR, abs/1706.09364, 2017. Tradução nossa. Disponível em: <<http://arxiv.org/abs/1706.09364>>.

Yang, W.; Jiachun, Z. Real-time face detection based on yolo. In: 2018 1st IEEE International Conference on Knowledge Innovation and Invention (ICKII). [S.l.: s.n.], 2018. p. 221–224. Tradução nossa.