UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Arthur Costa Serra

Análise de função de custo aplicadas no algoritmo Retinanet

São Luís - MA

2019

Arthur Costa Serra

Análise de função de custo aplicadas no algoritmo Retinanet

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Geraldo Braz Júnior

São Luís - MA

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Serra, Arthur Costa.
Análise de função de custo aplicadas no algoritmo Retinanet / Arthur Costa Serra. - 2019.
45 f.

Orientador(a): Geraldo Braz Junior.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2019.

1. Aprendizagem de máquina. 2. Classificação. 3. Detecção de objetos. 4. Função de custo. 5. Regressão. I. Braz Junior, Geraldo. II. Título.

Arthur Costa Serra

Análise de função de custo aplicadas no algoritmo Retinanet

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado em 20 de Dezembro de 2019: São Luís - MA

Prof. Dr. Geraldo Braz Júnior
Orientador
Universidade Federal do Maranhão

Prof. Dr. João Dallyson Sousa de Almeida
Membro da Banca Examinadora
Universidade Federal do Maranhão

Prof.ª Dr.ª Simara Viera da Rocha
Membro da Banca Examinadora
Universidade Federal do Maranhão

São Luís - MA
2019

Agradecimentos

Agradeço a minha mãe e minha irmã, que estiveram comigo o tempo todo.

Agradeço ao clã 1337, Pedro Henrique Carvalho Vieira, Sebastião Henrique Nascimento Santos e Eduardo Silva Vieira, amigos desde o ensino médio e para o resto da vida. Também a Layane Menezes Azevedo que desde o começo da graduação se tornou uma grande amiga.

Agradeço aos meus amigos e parceiros de laboratório João Vitor Ferreira França, José Eduardo de Souza Oliveira, Carlos Vinícios Martins Rocha, Paulo Renato Conceição Mendes, Pedro Vinicius Almeida de Freitas e Italo Fernandes Santos da Silva, com quem pude aprender muito.

Agradeço a Equatorial Energia e a todos envolvidos no projeto P&D Autoleitura, em especial a Simara Viera da Rocha.

Por último, mas não menos importante, agradeço ao meu orientador Geraldo Braz Júnior que sempre foi uma inspiração para mim e para meus amigos e por ser um exemplo enquanto profissional e ser humano.

"A sorte favorece os corajosos."

(Alexandre, o Grande)

Resumo

As funções de custo são avaliadores do aprendizado obtidos por meio da experiência. Este trabalho demonstra, através da rede neural de detecção de objetos Retinanet, como funciona e qual a importância das funções de custo no processo de aprendizado por meio da descida de gradiente. Evidenciando as funções de custo de classificação e regressão e como elas são mutuamente afetas no contexto de medidores de energia elétrica.

Palavras-chave: Função de custo, Detecção de Objetos, Classificação, Regressão, Aprendizagem de máquina.

Abstract

Cost functions are appraisers of learning gained through experience. This work demonstrates, through the Retinanet object detection neural network, how it works and how important cost functions are in the learning process through gradient descent. Highlighting the classification and regression cost functions and how they are mutually affected in the context of electricity meters.

Keywords: Loss Function, Object Detection, Classification, Regression, Machine Learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CE	Cross Entropy
CNN	Convolutional Neural Network
CR	Correntropy
EI	Expected Improvement
FL	Focal Loss
FPN	Feature Pyramid Network
HOG	Histograms Oriented Gradients
IoU	Intersection over Union
MACE	Minimum Average Correlation Energy
MAE	Mean Absolute Error
mAP	mean Average Precision
MLP	Multilayer Perceptron
MSE	Mean Squared Error
RPN	Region Proposal Network
SVM	Support Machine Vector
TPE	Tree-Structured Parzen Estimator

Sumário

1 Introdução

Aprendizagem de máquina é uma área da inteligência artificial que possibilita uma representação computacional da habilidade de aprendizado por meio da experiência (REN, 2018). A partir da década de 1950, diversas pesquisas vêm abordando algoritmos de aprendizagem de máquina, desde os algoritmos mais simples como o *multilayer perceptron* (MAHMOUDI et al., 2016), que busca representar computacionalmente o comportamento de uma rede de neurônios no processo de aprendizagem, até algoritmos mais complexos de aprendizagem profunda, como redes neurais convolucionais (CNNs). A aprendizagem de máquina atua no processo de otimização dos parâmetros em relação à capacidade de predizer ou estimar uma informação. Algoritmos de aprendizagem profunda são capazes de definir seus próprios parâmetros diretamente a partir dados e definem cada representação como combinações de outras e simplificadas (GOODFELLOW; BENGIO; COURVILLE, 2016). A profundidade é definida pelas representações sucessivas, remapeando os vetores de entrada para diversos espaços até atingir a representação desejada. Reduzindo o custo computacional, por receber um dado bruto como entrada, obsolescendo a rotulação e a extração de características antes do processo de estimação ou predição.

A viabilidade desses algoritmos depende diretamente das formas em que são treinados e as métricas utilizadas para validar o conhecimento adquirido. Tais métricas podem tanto estar focadas no percentual de acertos dos algoritmos quanto em fórmulas mais complexas que avaliam o desempenho de maneira mais robusta, considerando a possibilidade de concordância ocorrer. As funções de custo são métodos para avaliar o quanto uma abordagem modela um problema.

Mesmo que a escolha de uma função de custo possa parecer trivial, devido a frequente utilização das funções mais simples e já estabelecidas na literatura, essa seleção pode ser bem mais complexa dependendo do objetivo do problema, por haver uma grande diversidade de funções de custo, cada uma com sua particularidade de operação. Essas particularidades podem ser o desbalanceamento de classes (da base de dados ou na saída predita), ou presença de ruídos na base. Assim, os hiperparâmetros de treinamento ótimos podem variar de acordo com a função de custo utilizada, reforçando a importância da escola da função de custo antes da otimização dos hiperparâmetros.

1.1 Objetivos

O objetivo deste trabalho é demonstrar a importância que o papel das funções de custo possui no processo de aprendizado, tanto no contexto de classificação quanto no contexto de regressão. Aplicando associações de custo de classificação e regressão para

avaliar o aprendizado da rede neural de detecção de objetos Retinanet, com hiperparâmetros de treinamento definidos estaticamente.

1.1.1 Objetivos específicos

Para alcançar o objetivo de geral deste trabalho, faz-se necessário alcançar os objetivos específicos:

- Produzir uma metodologia de avaliação do impacto das funções de custo no processo de aprendizado;
- Adaptar a rotina de treinamento da rede neural Retinanet, de tal forma que otimize os hiperparâmetros funções de custo abordadas;
- Avaliar o impacto das funções de custo por meio de métricas de precisão de classificação e regressão na detecção de objetos em imagens de medidores de energia elétrica.

1.2 Trabalhos Relacionados

Esta seção apresenta alguns trabalhos disponíveis na literatura relacionados a proposição de funções de custo, comparando com abordagens mais tradicionais e em vários contextos de predição.

Ren (2018) propõe em seu trabalho uma implementação do algoritmo *Extreme Learning Machine*(HUANG; ZHU; SIEW, 2006), com uma função de custo baseado no princípio estatístico chamado *Correntropy*. Este princípio é uma medida de semelhança entre duas variáveis aleatórias, treinado por meio da otimização quadrática para lidar com a não convexidade do algoritmo ELM (*Extreme Learning Machine*). O trabalho proposto obteve bons resultados em relação a abordagens mais comuns na classificação como o SVM (*Support Machine Vector*). Embora tais resultados só apresentaram estabilidade em bases de dados extremamente ruidosas, como conjunto de dados de diagnóstico de câncer de mama em Wisconsin (*WDBC*) ou o conjunto de dados de clientes de distribuidores atacadistas (*Wholesale Dataset*) (DUA; GRAFF, 2017).

A *Correntropy* também foi abordada no trabalho de Singh, Pokharel e Principe (2014), que busca demonstrar que a função de custo baseada em *Correntropy* consegue se aproximar da função de perda ideal de 0–1, sem perder a convexidade. Utilizando-a para treinar uma rede neural MLP (*Multilayer Perceptron*) pode torna-la imune ao *overfitting* e mais robusta a *outliers*. Mantendo desempenho de generalização consistente e melhor em comparação com outras funções de perda comumente utilizadas, mesmo após treinamento prolongado.

Já no trabalho de Torre, Puig e Valls (2018) é demonstrado que a superioridade da função de custo *Weighted kappa*, na classificação e na regressão, em relação à função de custo logarítmico padrão. Balanceando o peso das classes na base de diagnóstico de retinopatia diabética, aplicando a função em uma rede MLP.

Frequentemente na literatura surge funções de custo novas que soluciona algum problema específico, proveniente dos dados ou dos algoritmos. Avaliando o desempenho das funções no contexto de classificação e regressão separadamente. Neste trabalho, veremos uma comparação de desempenho das funções de custo, em um ambiente no qual a classificação e a regressão coexistem no mesmo contexto de detecção de objetos.

1.3 Organização do trabalho

O restante deste trabalho está estruturado da seguinte forma:

O Capítulo 2 trata da fundamentação teórica das funções de custo e do algoritmo utilizado nesse estudo. São abordados conceitos referentes as funções de custo de classificação e regressão, redes nerais, redes neurais convolucionais e redes neurais de detecção de objetos.

O Capítulo 3 evidência as etapas que compõem a metodologia proposta para este trabalho. Tais como a descrição da base de dados, a definição do algoritmo e os demais hiperparâmetros, definição da combinação de funções de custo, otimização das funções de custo parametrizadas e, por fim, avaliação das métricas finais de aprendizado.

O Capítulo 4 disserta sobre os resultados obtidos em relação desempenho das funções de custo nos experimentos de detecção e reconhecimento de objetos, em imagens de medidores de energia elétrica.

Por fim, o Capítulo 5 apresenta as considerações finais sobre os resultados.

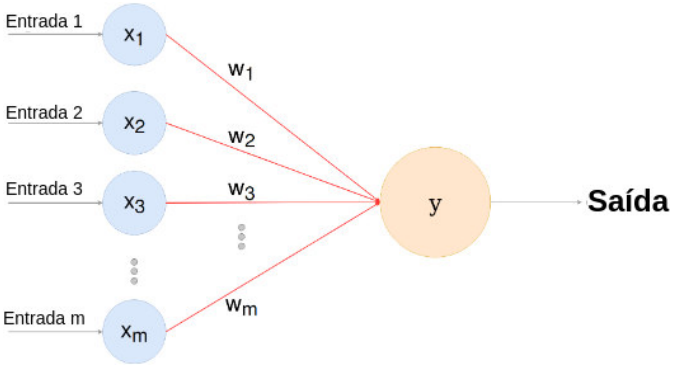
2 Fundamentação Teórica

Neste capítulo são apresentados os principais fundamentos de redes neurais, redes neurais convolucionais, detecção de objetos e funções de custo utilizados para construção da metodologia proposta nesta.

2.1 Redes Neurais

Biologicamente, um neurônio é um tipo de célula presente no sistema nervoso com a função de transmitir informação através de impulsos elétricos. Análoga ao sistema nervoso, uma rede neural artificial é composta por um conjunto de neurônios artificiais, chamados de *perceptron*, desenvolvido para resolver diversos problemas de inteligência artificial. Proposto inicialmente por Rosenblatt (1961), um *perceptron* é composto por uma camada de entrada, uma camada de pesos, uma camada de saída como na Figura 1.

Figura 1 – Representação de um Perceptron



Fonte: Adaptado de (ARUNAVA, 2018a)

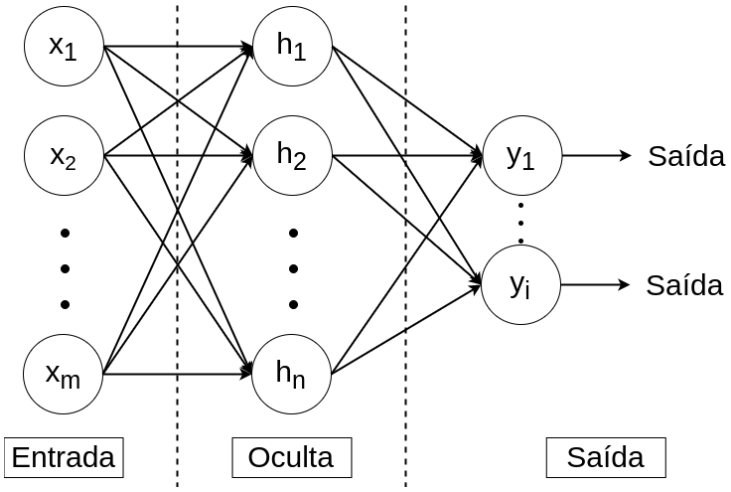
A camada de entrada recebe x_m valores reais, que são ponderados pelas sinapses w_m, cada uma com um peso associado e uma camada de saída y que calcula uma pontuação com base nas entradas e nos pesos com base na Equação 2.1; por fim estimar a pontuação através de uma função de ativação. Uma função de ativação introduz um conceito de não linearidade na rede, por tanto uma rede neural sem uma função de ativação, independente da profundidade, só resolve problemas lineares (ARUNAVA, 2018a).

$$f(\sum_{i=0}^{m} x_i w_i) \tag{2.1}$$

Um único *perceptron* não consegue resolver problemas não lineares. Em vista disso, uma rede MLP (*Multilayer Perceptron*) adiciona mais uma camada ao *perceptron*, chamada

camada oculta. Com exceção da camada de entrada, todos os nós da rede possuem uma função de ativação não linear associada.

Figura 2 – Multilayer perceptron



Fonte: Autor

O ajuste dos pesos de uma rede neural pode ser dado pela descida do gradiente. A descida do gradiente é um algoritmo de otimização usado para encontrar os melhores valores para os pesos que minimizem a função de custo. O processo de descida começa com valores normalizados aleatórios para os pesos; em seguida, é calculado o custo associado a esses pesos, para então calcular o gradiente. O gradiente, nada mais é que, a derivada da função de custo no ponto referente aos pesos, para conhecer sua direção de descida e ajustar os pesos para o mínimo da função de custo (Figura 3). Para atualizar os pesos, uma taxa de aprendizado (*learning rate*) deve ser definida para saber o quanto os pesos podem mudar a cada iteração como definido na Equação 2.2, onde lr define a taxa de aprendizado e δ a derivada do custo associado (BROWNLEE, 2016a).

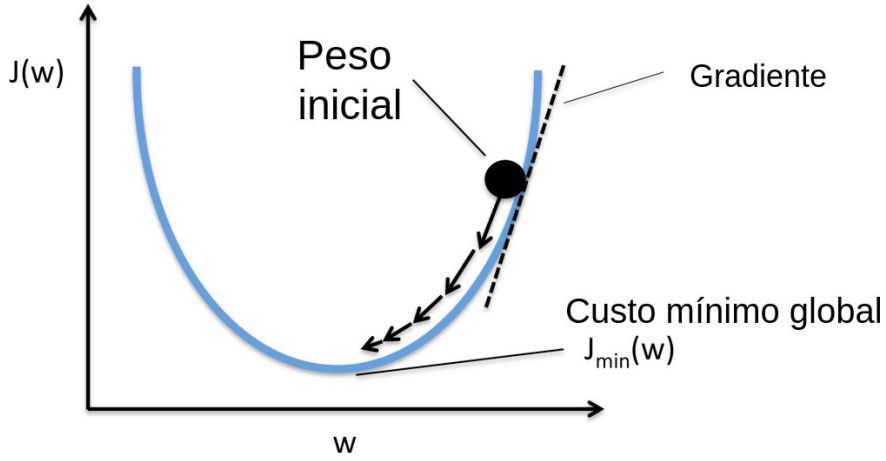
$$peso = peso - (lr * \delta) \tag{2.2}$$

No processo de descida pode ocorrer o acumulo de gradiente nas iterações resultando em um valor muito alto e gerando grandes mudanças no ajuste dos pesos e dificultando o aprendizado. Esse fenômeno é conhecido com explosão de gradiente e pode ser resolvido com uma técnica muito simples, conhecida como recorte, que define uma norma (*clipnorm*) ajustando o valor do gradiente para que não escape da curva de descendência do custo (BROWNLEE, 2016b).

2.1.1 Rede Neural Convolucional

Rede neural convolucional (CNN ou *ConvNets*) é um tipo de rede neural profunda usada para solucionar problemas de visão computacional, tais como classificação e detecção

Figura 3 – Descida de gradiente

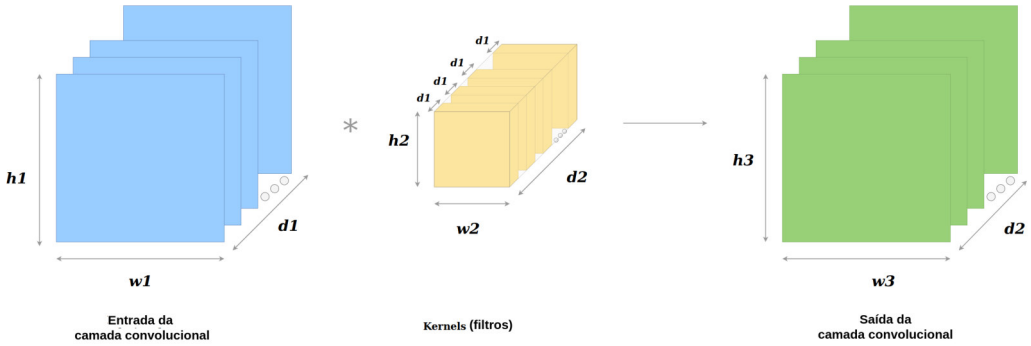


Adaptado de (ACADEMY, 2019)

de objetos em imagens. Em suma, uma CNN possui três tipos de camadas: convolucional, *pool* e totalmente conectada.

A camada convolucional (Figura 4) possui uma camada de entrada de dimensão $[h_1 * w_1 * d_1]$, onde $[h_1 * w_1]$ é a dimensão da imagem e d_1 a quantidade de canais, essa entrada passa por d_2 filtros $[h_2 * w_2 * d_1]$, que é passado em todos os canais multiplicando a cada posição do filtro na matriz inicial e somando para a posição correspondente das d_2 matrizes resultantes de dimensão $[h_3 * w_3]$.

Figura 4 – Camada convolucional

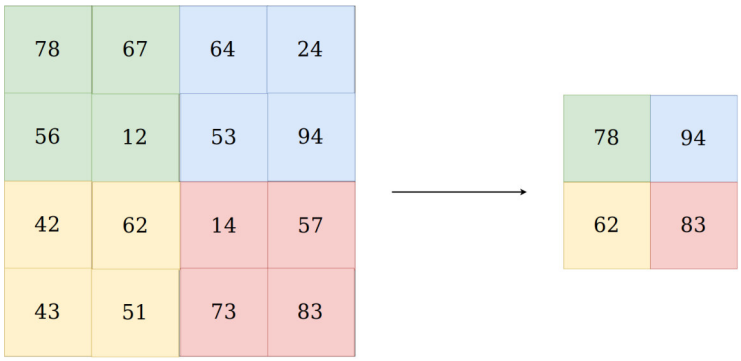


Fonte: Adaptado de (ARUNAVA, 2018b)

A camada *pool* objetiva reduzir o número de parâmetros de entrada do tensor para reduzir o sobre-ajuste, extrair características mais representativas e, assim, exigir menos recurso computacional. Existem dois tipos de camada *pool*, o *Max pooling* e o *Average pooling*. No caso do *Max pooling*, como demonstrado na Figura 5, a cada quadro do tamanho $[n * n]$ da matriz de entrada é extraído o valor máximo desse mesmo contexto

para uma matriz resultante. No caso do *Average pooling*, é extraído a média do quadro. Esse processo é aplicado em todos os canais da matriz de entrada.

Figura 5 – Camada *Max Pooling*



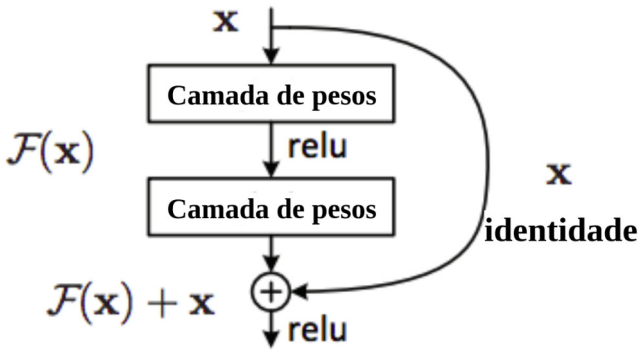
Fonte: (ARUNAVA, 2018b)

Por fim, a camada totalmente conectada é uma rede neural com estrutura e funcionamento análogos ao de uma MLP, que recebe como entrada para posição da matriz resultante da camada *pool* (ARUNAVA, 2018b).

Com o tempo, surgiram diversas CNNs com diferentes estratégias de combinações de camada, deixando as redes mais profundas, evitando o uso frequente de camadas pool para preservar a resolução da matriz de entrada e aprofundar a rede cada vez mais, sendo que cada camada convolucional se especializa em detectar algum tipo específico de característica. Contudo, segundo He et al. (2016), existe uma profundidade máxima que essa abordagem pode alcançar (56 camadas) e mesmo assim, quanto mais profundo, mais complexo o processo de aprendizagem é devido ao aumento de chances de ocorrer sobre-ajuste e explosão de gradiente.

A maneira proposta por He et al. (2016) para solucionar os problemas causados por uma grande profundidade das redes convolucionais são os blocos residuais (Figura 6). Seja x uma matriz de entrada de um conjunto de camadas convolucionais $F(x)$, um bloco residual consiste em uma adição da entrada do bloco com saída das convoluções. Porém, nem sempre a dimensão de x é a mesma de $F(x)$. Para esses casos a adição é feita a partir da projeção linear de ambas as matrizes. Dessa forma, possibilitou-se a construção de modelos bem mais profundos, como as redes Resnets de até 152 camadas (HE et al., 2016). O aumento da profundidade deve aumentar a precisão da rede, desde que a imagem de entrada seja suficientemente grande para que o sinal chegue ao final da rede, com valor significativo para o ajuste de todas as camadas.

Figura 6 – Bloco residual

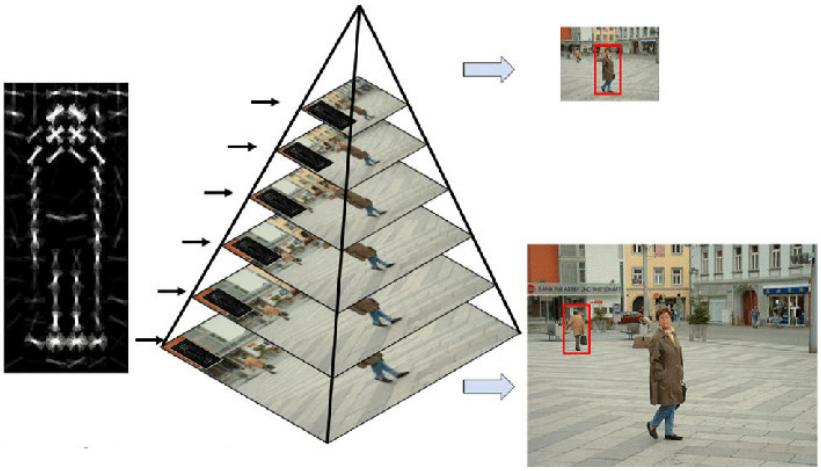


Adaptado de (HE et al., 2016)

2.1.2 Rede Neural de Detecção de Objetos

Detecção de objetos é um problema de visão computacional que consiste em identificar objetos de determinadas classes em uma imagem com um contexto qualquer de diversas maneiras, desde da simples construção de uma *bounding box* em volta do objeto até a segmentação de todos os pixeis referentes a este. Embora atualmente quase todas as soluções envolvam CNNs, o problema já vem sendo abordado antes da popularização das redes neurais.

Figura 7 – Detecção de objetos tradicional.



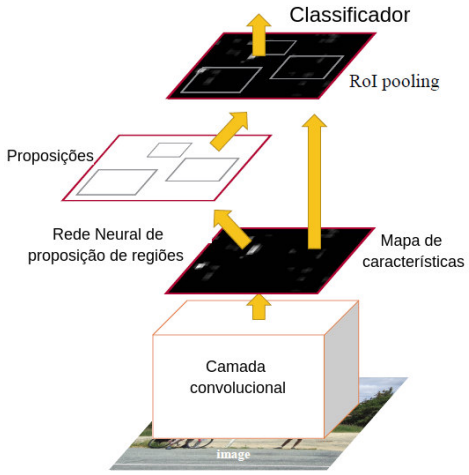
Adaptado de (GANESH, 2019)

Como observado na Figura 7, dado um *template* de características, extraído por um método como o HOG (*Histograms Oriented Gradients*) (DALAL; TRIGGS, 2005), é buscado em diversas dimensões de uma imagem de entrada, características semelhantes (GANESH, 2019).

As redes neurais de proposição de região, ou RPN (Region Proposal Network) (REN et al., 2017a), representa na Figura 8, recebem um mapa de característica de uma

CNN qualquer, sem a camada totalmente conectada, e identificam as possíveis regiões que contenham objetos, para em seguida as classificar cada uma delas para saber a quais objetos representam. Contudo, gerar regiões com formatos variados é extremamente improvável

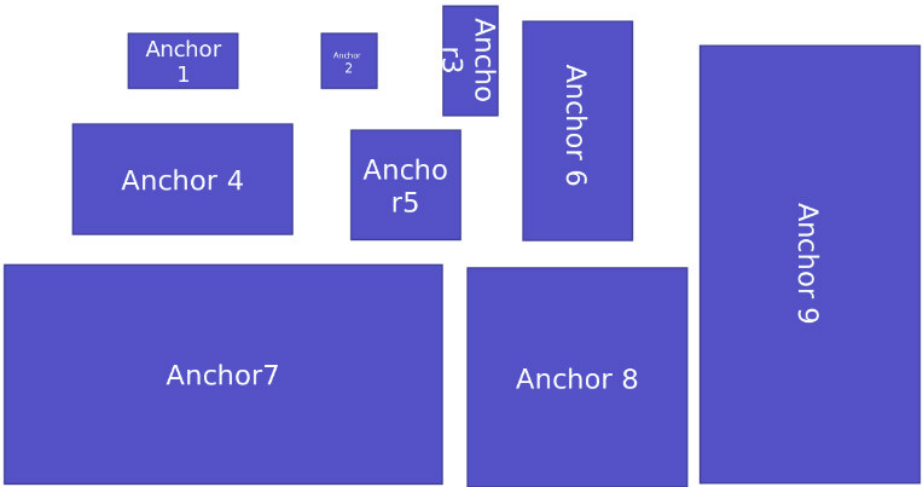
Figura 8 – Detecção de objetos em dois estágios.



Fonte: Adaptado de (REN et al., 2017b)

com convoluções padrão, por conta disso, usa-se *Anchor boxes* (Figura 9) que é uma série de caixas delimitadoras de diversos formatos e tamanho para servir de modelo para a busca de objetos nas imagens (REN et al., 2017b). Assim, para cada *Anchor Box* há duas saídas, uma de regressão que prediz às coordenadas dos extremos inicial e final (x_0, y_0, x_i, y_i) e uma de classificação que determina a probabilidade da presença de um objeto estar ou não na região proposta (JAY, 2018).

Figura 9 – Anchor Boxes.



Fonte: (JAY, 2018)

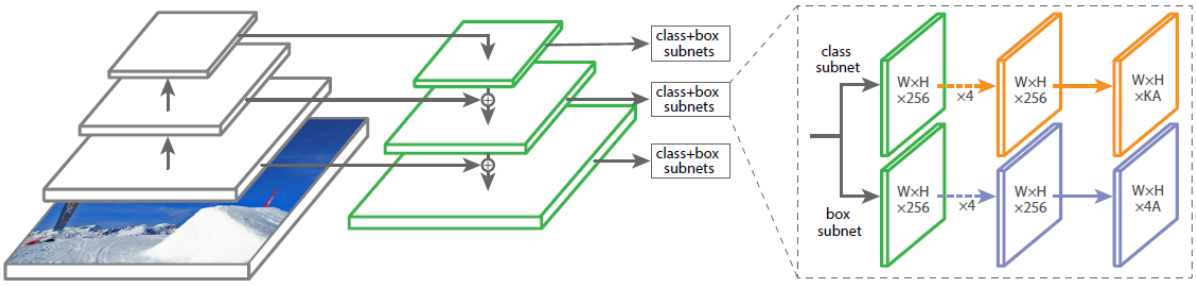
Os principais problemas das redes RPN são a variedade de escala dos objetos,

mesmo com diversos formatos de âncoras, e a performance de inferência, por precisar primeiro gerar possíveis regiões para, só depois, classificar qual é o objeto.

Para solucionar o problema de escala, utiliza-se redes FPN (Feature Pyramid Network) (LIN et al., 2017a). Uma FPN extrai vários mapas de características, cada um em uma dimensão diferente, para uma única imagem de entrada e aplica supressão não-máxima sobre todas as *Anchor Boxes*. O problema de performance precisa de uma abordagem mais integrada de detecção; os chamados detectores de um estágio, que requer só uma passagem da imagem pela rede para detectar todos os objetos.

O problema dos detectores de um estágio está, justamente, na proposição das regiões de possíveis objetos. Uma RPN gera, em média, duas mil possíveis regiões por imagem; uma rede de um estágio, como a Retinanet, gera até 100 mil possíveis regiões. Isso pode tornar o treinamento mais complexo, por causa da grade densidade de instâncias falso positivas para a presença de objeto. Uma rede Retinanet é composta por uma FPN para extração de característica chamada de *backbone*; uma sub-rede de classificação que prevê a probabilidade da presença de um objeto para todas as âncoras A em K classes; uma sub-rede de regressão que prediz o deslocamento das *bounding boxes* para o objeto mais próximo e possui comportamento idêntico ao da sub-rede de classificação exceto pelas saídas lineares $[x_0, y_0, x_i, y_i]$ para cada âncora A. Cada sub-rede opera em todas as dimensões da FPN (LIN et al., 2017b). Para que uma rede Retinanet torne-se mais precisa o avaliador de aprendizado é vital, para que consiga descartar a enorme carga de falsos positivos.

Figura 10 – Arquitetura Retinanet.



Fonte: (LIN et al., 2017b)

A seção seguinte apresenta algumas das funções de custo mais utilizadas na literatura.

2.2 Funções de custo

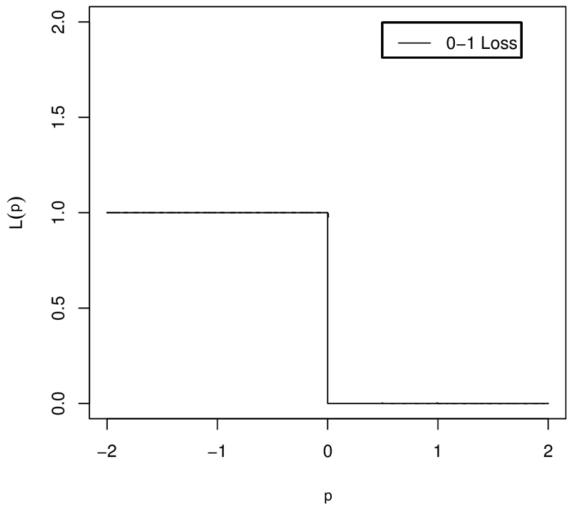
O custo é definido essencialmente como a diferença calculada a partir da previsão de uma função $f(p)$ em relação a um valor esperado $\hat{p}$. O procedimento de otimização

partindo do custo é conhecido como Minimização de Risco Empírico, princípio amplamente utilizado para classificação. A função de custo mais básica é a função 0–1 (Equação 2.3), em que, para um determinado conjunto predito, p só será ótimo quando for igual a um conjunto real $\hat{p}$.

$$L(p, \hat{p}) = \begin{cases} 0 & se \quad p = \hat{p} \\ 1 & se \quad p \neq \hat{p} \end{cases} \tag{2.3}$$

Embora seja uma função de custo ótima, é computacionalmente inviável para otimização em modelos probabilísticos por ser não-continua e não-convexa (Figura 11), sem suavização do espaço de busca que impede uma convergência de aproximação, tendendo sempre ao sobre-ajuste do algoritmo. Sendo assim a solução é utilizar funções de custo convexas que possam se aproximar da estimação da função 0–1 (BISHOP, 2006).

Figura 11 – Curva de representação da função de custo 0–1



Fonte: Autor

Uma função é dita convexa em um intervalo $[a, b]$, se a região sobre a curva for um conjunto convexo, ou seja, a corda que ligar quaisquer dois pontos acima da curva esteja totalmente dentro do conjunto. Isso equivale dizer que, $\forall(x, y) \in [a, b]$ e $\forall t \in [0, 1]$ a função deve obedecer à Equação 2.4 (BOYD, 2011):

$$f(tx + (1 - t)y) \leq tf(x) + (1 - t)f(y) \tag{2.4}$$

2.2.1 Função de Custo de Classificação

Funções de custo de classificação são modeladas para os problemas nos quais, dado uma entrada x o objetivo é obter a probabilidade de pertencer a um conjunto

preestabelecido de rótulos y. A função de custo deve minimizar a margem de erro para um conjunto finito de rótulos.

2.2.1.1 Cross Entropy

Na teoria da informação, é preciso tratar matematicamente a medida das informações para melhor dinâmica de transmissão em um canal binário. Para tal campo de codificação é usado o conceito de entropia para definir a quantidade de bits necessários para representar um dado, basicamente mensurando o quão incerto é o valor de uma variável aleatória ou a saída de um processo aleatório.

Segundo Shannon (1948), dado um sistema $S = \{x_1, x_2, ..., x_n\}$ em que $p_i = \frac{1}{n}$ é a probabilidade associada a x_i, a entropia H pode ser definida pela Equação 2.5.

$$H^S(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \ log_2(p_i) \tag{2.5}$$

No contexto de aprendizagem de máquina, o conceito de entropia precisa ser um pouco mais amplo. Já que surge a necessidade de mensurar a entropia entre dois sistemas, estimado p e esperado $\hat{p}$. Para tal, a fórmula da entropia dada pela Equação 2.5 dá lugar a entropia cruzada, ou *Cross Entropy* (CE).

A diferença entre CE e entropia é chamado divergência de Kullback-Leibler ou divergência KL (Equação 2.6) demonstra que, quanto mais semelhantes forem as distribuições p_i e $\hat{p}_i$ menor será a divergência entre CE e H (RICHARD, 2013).

$$\begin{aligned} D_{KL}(CE, H) &= CE(p_i, \hat{p}_i) - H(\hat{p}_i) \\ &= \textstyle\sum_{i=1}^{n} H(\hat{p}_i) \ \cdot \ [log(H(\hat{p}_i)) - log(CE(p_i, \hat{p}_i))] \end{aligned} \tag{2.6}$$

A *Cross Entropy*, enquanto função de custo de classificação, leva em consideração um conjunto de distribuição de tamanho N que representa o total de classes de um determinado problema. Isso transformando a *Cross Entropy* (Equação 2.7) em uma média ponderada e a distribuição p, em um vetor de tamanho N com as probabilidades para cada classe respectivamente nas posições dos vetores. Já a distribuição $\hat{p}$, como representa a distribuição esperada, recebe apenas os valores 0 e 1, sendo 1 para a posição da classe esperada e zero para as demais posições.

$$CE(p_i, \hat{p}_i) = -\frac{1}{N} \sum_{i=1}^{N} \hat{p}_i \cdot log(p_i) \tag{2.7}$$

Onde $CE(p_i, \hat{p}_i) = CE(\hat{p}_i, p_i)$, se e somente se, $p_i = \hat{p}_i$ resultando em uma estimativa perfeita e $CE(p_i, \hat{p}_i) = H(\hat{p}_i)$.

Como demonstrado por Lin et al. (2017b), a entropia cruzada tem uma forma simples de abordar o desiquilíbrio de classes basicamente introduzido um fator α no cálculo.

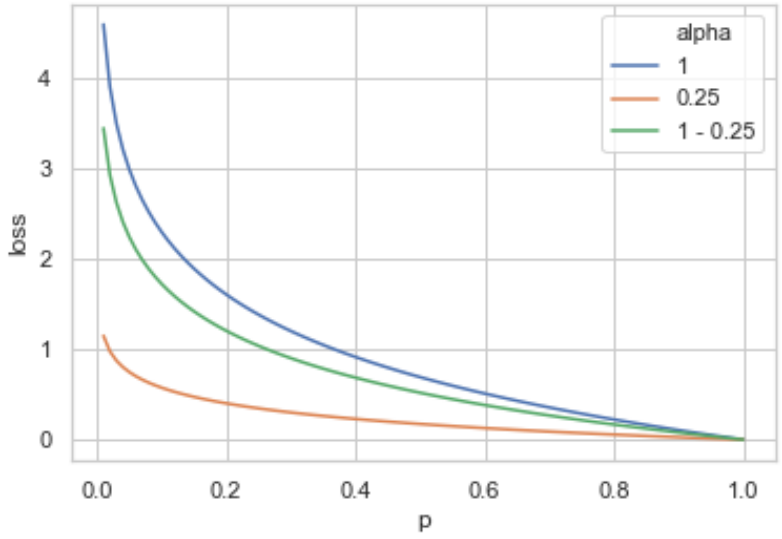
Sendo $\alpha \in [0, 1]$ a distribuição é definida por:

$$\alpha_t = \begin{cases} \alpha & se\ \hat{p} = 1 \\ 1 - \alpha & caso\ contrario \end{cases} \tag{2.8}$$

Desta forma, uma única função pode possuir mais de uma curva de representação, dando prioridade ao aprendizado de uma classe específica. Por exemplo, na Figura 12 podemos observar 3 curvas de representação, sendo a azul a curva padrão da entropia cruzada com $\alpha = 1$, e as outras duas curvas representam a Equação 2.9, que da mais prioridade de aprendizado para os casos onde $\hat{p} = 1$. Quando mais próximo dos eixos, mais precisa é a função.

$$CE(p_t) = -\alpha_t\ log(p_t) \tag{2.9}$$

Figura 12 – Curva de representação da função de custo CE



Fonte: Autor

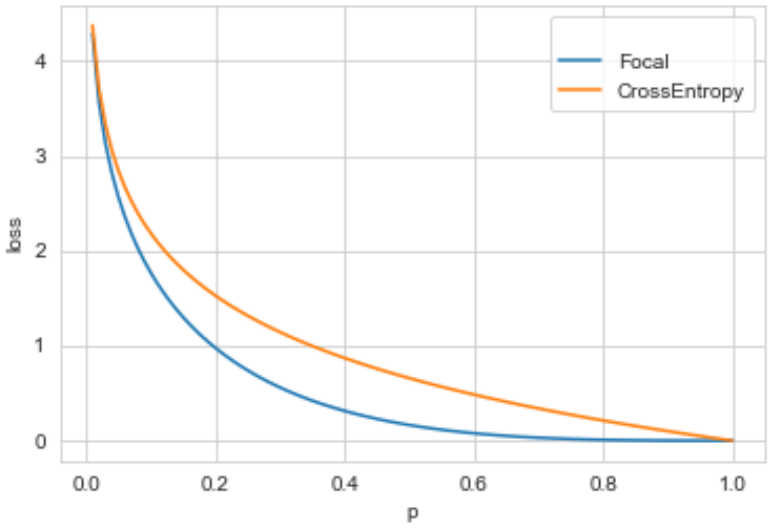
2.2.1.2 Focal Loss

A *Focal Loss*, essencialmente, é uma adaptação da CE objetivando diminuir o peso do custo atribuído a instância com fácil classificação. A abordagem evita que o algoritmo classificador se sobrecarregue com instâncias negativas fáceis e concentra o aprendizado em exemplos mais heterogêneos da classe verdadeira. O principal problema da Entropia Cruzada com balanceamento é o grande desbalanceamento das instâncias fáceis em relação as difíceis, porque mesmo com α equilibrando positivos e negativos, as instâncias fáceis domina totalmente a descida do gradiente de otimização.

Diante disso, Lin et al. (2017b) propuseram a Equação 2.10, um parâmetro de foco γ parte do fator $(1-p_t)^\gamma$. Quando uma instância é classificada incorretamente é atribuído um p_t pequeno a ela, quanto mais p_t é próximo de 1 mais fácil é a instância. O fator $(1-p_t)^\gamma$ será sempre maior quando as instâncias forem difíceis. Quando $\gamma = 0$ o custo focal equivale a CE, a medida em que γ aumenta maior o efeito do fator de foco (Figura 13).

$$FL(p_t) = -\alpha_t(1-p_t)^\gamma log(p_t) \tag{2.10}$$

Figura 13 – Curva de representação da função de custo FL



Fonte: Autor

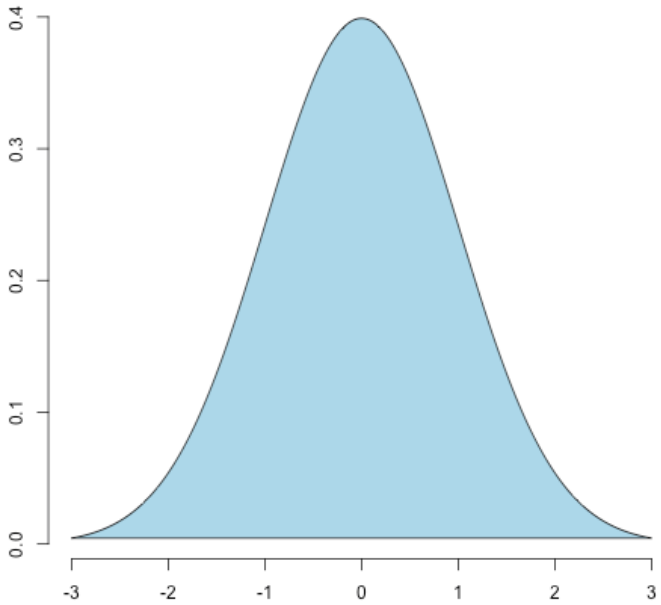
2.2.1.3 Correntropy

Define-se *Correntropy* a medida estatística de similaridade entre duas ou mais variáveis aleatórias, integrando uma função de densidade. Comumente utilizado em problemas como regressão robusta, filtragem adaptativa, detecção de afinação no discurso, MACE (*Minimum Average Correlation Energy*), filtragem para reconhecimento de objetos, etc (SINGH; POKHAREL; PRINCIPE, 2014). Para classificação utilizou-se a distribuição normal de densidade, ou kernel Gaussiano (Figura 14). O é kernel definido na Equação 2.11. Onde z é uma dada distribuição e σ um desvio padrão arbitrário maior que zero.

$$\kappa_\sigma(z) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{\|z\|^2}{2\sigma^2}\right) \tag{2.11}$$

Haja vista duas distribuições p e $\hat{p}$ desconhecidas, a *Correntropy* pode ser medida a partir da Equação 2.12. Diferente do CE, a medida de similaridade abordada possui

Figura 14 – Distribuição normal de densidade



Fonte: Autor

propriedade simétrica, ou seja, $CR(p, \hat{p}) = CR(\hat{p}, p)$ para uma determinada distribuição de tamanho N (REN, 2018). A utilidade da largura do kernel σ em relação a outras medidas de semelhança é sua robustez a dados ruidosos na base de aprendizado, já que o comportamento da distribuição tende a acumular na maioria de dados corretamente anotados.

$$CR(p, \hat{p}) = \frac{1}{N} \sum_{i=1}^{N} \kappa_\sigma(\hat{p}_i - p_i) \tag{2.12}$$

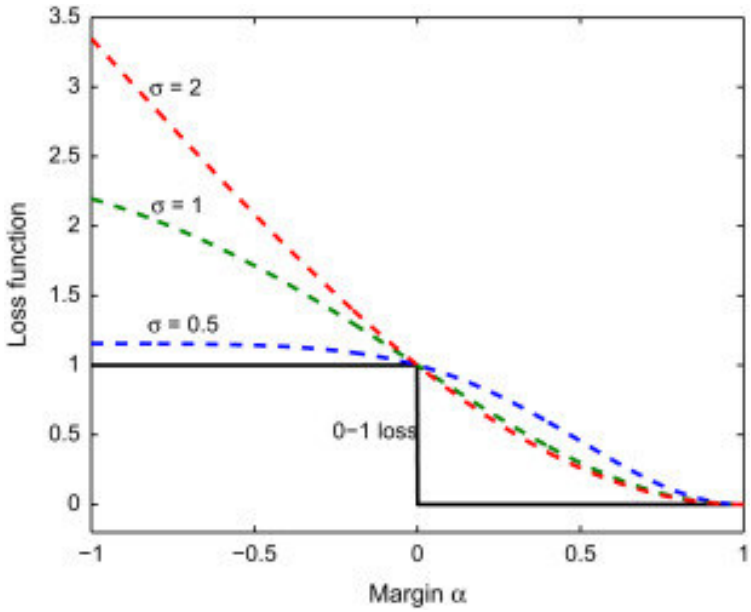
Na classificação o objetivo é maximizar a semelhança entre um conjunto predito e um conjunto esperado. Para isso, a função de custo deve minimizar o risco de tal modo que a seja equivalente à maximização da *Correntropy*, definindo a Equação 2.13 como o risco associado à semelhança (Equação 2.12) ponderado por uma variável $\alpha > 0$ arbitrária. Quanto maior for a largura do kernel, mais importância será dada a instâncias mal classificadas. Observando a representação da função na Figura 15, pode-se notar sua aproximação à função ótima 0-1 sem perder a convexidade de suavização do gradiente.

$$CR_{Cost}(p, \hat{p}) = \alpha \, (1 - \frac{1}{N} \sum_{i=1}^{N} \kappa_\sigma(\hat{p}_i - p_i)) \tag{2.13}$$

2.2.2 Função de Custo de Regressão

Matematicamente, regressão é o processo de obtenção de uma função de um tipo predefinido que melhor represente a dependência entre duas variáveis. Por isso, funções de

Figura 15 – Curva de representação da função de custo CR_{Cost}



Fonte: (SINGH; POKHAREL; PRINCIPE, 2014)

custo de regressão têm um objetivo mais complexo que o de classificação, dado que uma entrada x pertence a um conjunto contínuo y.

2.2.2.1 Mean Absolute Error

A mais simples função de custo de regressão utilizada é a *Mean Absolute Error* (MAE). Também conhecida como função L1, em suma, visa minimizar a média soma das diferenças absolutas entre dado conjunto p predito e um conjunto esperado $\hat{p}$ (SAMMUT; WEBB, 2010a).

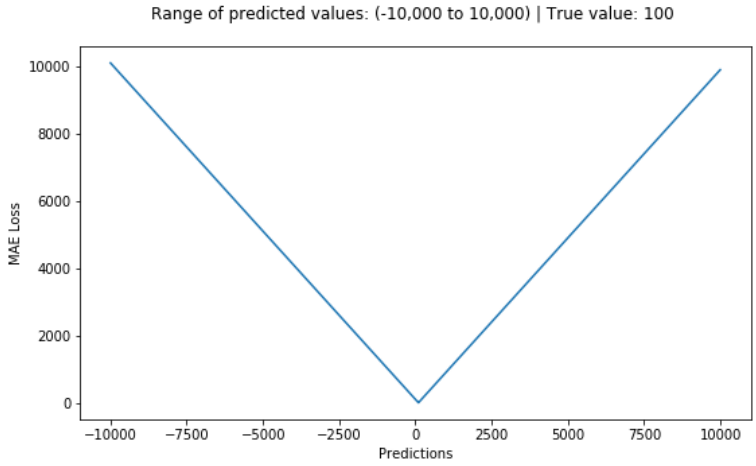
$$MAE = \frac{1}{N}\sum_{i=1}^{N} \|\hat{p}_i - p_i\| \tag{2.14}$$

A grande vantagem da utilização de função L1 é sua robustez para com dados anotados erroneamente ou extremamente distintos de seus semelhantes. O custo absoluto pode ser observado como a mediana de um conjunto, onde valores extremos não afetam tanto no resultado final. O maior problema para o aprendizado é a descida do gradiente, já que o gradiente, para esta função, não varia e será alto para valores bem próximos do mínimo global e consequentemente deixando o ajuste menos preciso ao final do treinamento (GROVER, 2018).

2.2.2.2 Mean Squared Error

A mais frequentemente usada função de custo de regressão é a *Mean Squared Error* (MSE), ou simplesmente função L2 (Equação 2.15). Ela é definida como a média do

Figura 16 – Cuva de representação da perda absoluta



Fonte: (GROVER, 2018)

somatório dos quadrados da diferença entre um conjunto predito p e um conjunto esperado $\hat{p}$ (SAMMUT; WEBB, 2010b).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{p}_i - p_i)^2 \tag{2.15}$$

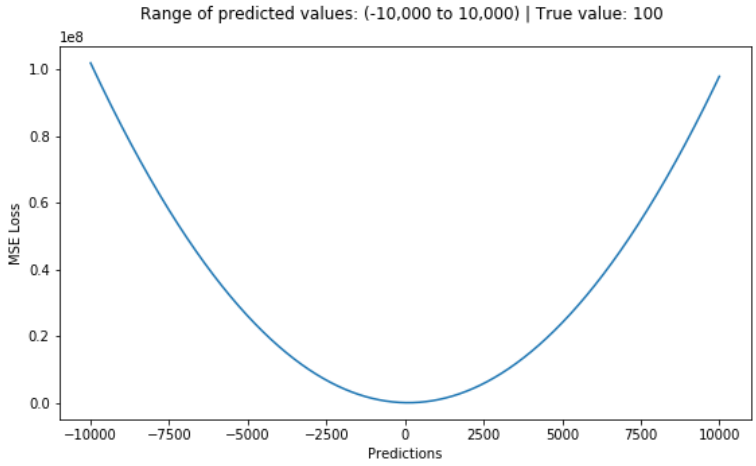
A principal vantagem da função L2 em comparação com a função L1 é sua capacidade de reduzir a magnitude do gradiente à medida que o algoritmo aprende devido a sua convexidade, permitindo um aprendizado mais preciso e confiável (Figura 17). Já que essa função pode ser observada como a média de um conjunto, torna-se extremamente sensível a elementos discrepantes no conjunto, o que a torna pouco prática para dados em que não há total garantia de que não haja valores muito distintos em relação aos seus semelhantes ou anotados de forma errônea (GROVER, 2018).

2.2.2.3 Smooth-L1

De modo a conservar as características boas presentes nas funções MAE e MSE, a *Smooth-L1* precisa manter a estabilidade de aprendizado no início do treinamento e a robustez às amostras discrepantes como na MAE e a precisão de convergência presente na MSE. Para tal, basta unir às duas funções de tal forma, que a partir de determinado ponto a curva de convergência alterne da função L1 para L2.

O ponto de união entre às duas funções é obtido onde o gradiente da MSE é igual ao gradiente da MAE. Seja $x = \hat{p} - p$, um conjunto arbitrário, a constante de suavização das duas funções é dado pelo valor de x cuja derivada de L1 seja igual à derivada de L2

Figura 17 – Cuva de representação da perda quadrática



Fonte: (GROVER, 2018)

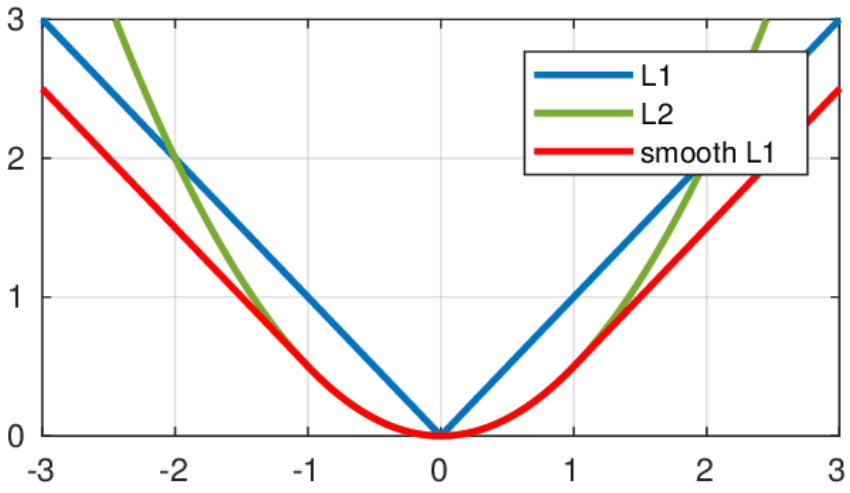
(Equação 2.16).

$$\begin{aligned} \frac{\mathrm{d}|x|}{\mathrm{d}x} &= \frac{\mathrm{d}x^2}{\mathrm{d}x} \\ 1 &= 2x \\ 1 &= 2\tfrac{1}{2} \\ 1 &= 1 \end{aligned} \tag{2.16}$$

Logo, temos a Equação 2.17:

$$Smooth_{L1}(x) = \begin{cases} \frac{1}{2}x^2 & se\ |x| < 1 \\ |x| - \frac{1}{2} & caso\ contrario \end{cases} \tag{2.17}$$

Figura 18 – Cuva de representação da perda L1 suavizada



Fonte: (FENG et al., 2017)

Já a *Smooth-L1* enquanto função de custo é dado pela média do somatório da curva suavizada em função da diferença $\hat{p} - p$ (Equação 2.18).

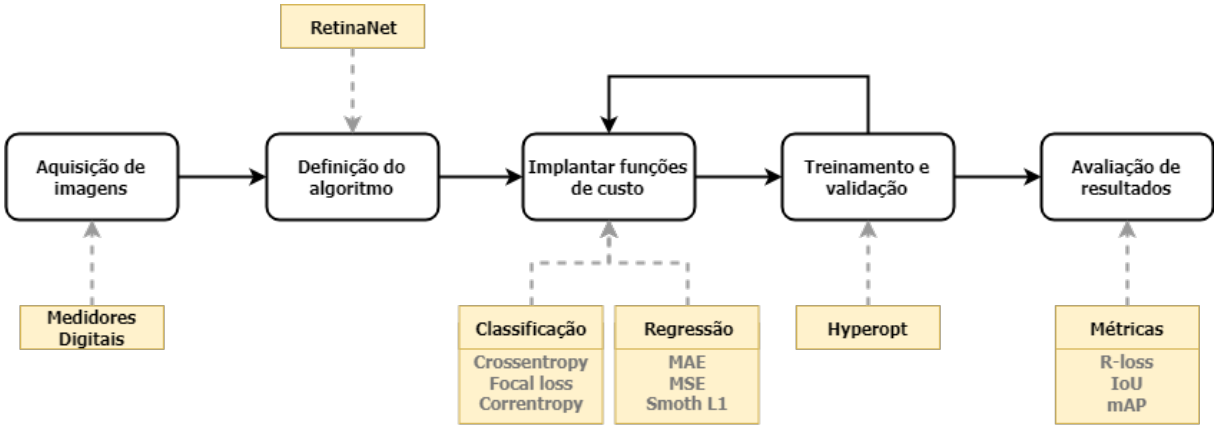
$$Smooth_{Cost}(p, \hat{p}) = \frac{1}{N} \sum_{i=1}^{N} Smooth_{L1}(x) \tag{2.18}$$

Neste capítulo foi apresentado noções sobre redes neurais e funções de custo. O próximo capítulo descreve uma metodologia de avaliação das funções de custo.

3 Metodologia

A metodologia para a demonstração do impacto das funções de custo na avaliação final de algoritmos de aprendizagem de máquina está representada na Figura 19 em um processo de cinco etapas descritas nas demais seções deste capítulo. Neste processo o ciclo entre as etapas de treinamento e implantação das funções de custo objetiva realizar todas as combinações possíveis de funções.

Figura 19 – Fluxograma de aplicação das funções de custo.



Fonte: Autor

3.1 Aquisição de Imagens

Nesse trabalho, foi utilizada uma base de dados com imagens de medidores de energia elétrica digitais, cedidas no âmbito do projeto de P&D Autoleitura com a Equatorial Energia. A base conta com uma diversidade de modelos digitais, como demonstra a Figura 20, mas todas as imagens possuem a mesma dimensão (1280 pixeis de altura por 960 pixeis de largura).

A base de medidores digitais possui um total de 3873 imagens que posteriormente foi dividida seguindo a proporção de 70% (2711 imagens) para treino e e 30% (1162 imagem) para teste.

Visando o objetivo de detecção de objetos cada grupo possui as anotações das coordenadas das *bounding boxes*, que são as coordenadas dos extremos de um retângulo, e sua respectiva classe de todos os objetos de interesse em cada imagem. Assim como representado na Figura 21, os objetos de interesse marcados nas imagens variam desde

Figura 20 – Exemplificação da diversidade de modelos de medidores

Fonte: Imagens base Autoleitura

medidor e display até cada um dos dígitos de 0 a 9. A quantidade de anotações por classe das bases analógica e digital então evidenciadas na Tabela 1.

Tabela 1 – Quantidade de anotações por classe

	Imagens											
	0	1	2	3	4	5	6	7	8	9	Display	Medidor
Treino	5835	1203	1007	879	798	706	758	643	1399	616	2675	2423
Teste	2441	533	426	402	339	335	286	273	586	281	1139	1033

Figura 21 – Representação das *bounding boxes* anotada

Fonte: Autor

3.2 Definição do Algoritmo

A etapa de definição do algoritmo configura a rede neural de detecção de objetos Retinanet (LIN et al., 2017b), abordada mais detalhadamente no Capítulo 2. De modo a melhor avaliar o efeito da variação dos resultados provocado pelas funções de custo, os hiperparâmetros da rede foram definidos de tal forma que não sejam modificados nas etapas subsequentes de otimização.

Utilizando como rede *backbone* de extração de característica a Resnet (HE et al., 2016) em sua versão reduzida com 50 camadas. O detector de objetos Retinanet é inicialmente compilado apenas com o método de otimização estocástica Adam (KINGMA; BA, 2014), com taxa de aprendizado (*learning rate*) 10^{-5} e recorte de gradiente (*clipnorm*) 10^{-3}. Posteriormente, o modelo é treinado por 20 épocas com 1356 passos de iterações, que equivale à metade do conjunto de treino, e lote de treinamento 2 para garantir que a cada época todo o conjunto de treino seja contemplado.

3.3 Implantar Funções de Custo

Essa etapa do processo define duas funções de custo, sendo uma para regressão e outra para classificação e seus respectivos parâmetros. Se necessário, cabe a esse processo testar todas as funções, cruzando todas as de classificação com as de regressão pré-estabelecidas na Seção 2.2. Todas as funções de custo descritas são mais frenquentemente utilizadas na literatura, por demonstrarem bom desempenho em relação a desbalanceamento e/ou ruídos (problemas presente na base de medidores).

Apenas as funções de classificação utilizadas nesse experimento são parametrizadas. As de funções de regressão têm lógica única, sem variações de parâmetro. No instante zero desta etapa, cada parâmetro são iniciados com um valor real aleatório que respeite todas as restrições de intervalo:

- Crossentopy loss.
  - Alpha: intervalo real de 0 a 1.
- Focal loss.
  - Alpha: intervalo real de 0 a 1.
  - Gamma: intervalo inteiro de 0 a 5.
- Correntropy loss.
  - Alpha: intervalo inteiro de 0 a 5.
  - Sigma: intervalo real de 0 a 1.

3.4 Treinamento e Validação

Após todo processo de compilação do modelo, inicia-se o primeiro treinamento. As métricas de custo para a classificação e regressão são avaliadas pela função de minimização da biblioteca *hyperopt* (BERGSTRA; YAMINS; COX, 2013), que no que lhe concerne utiliza o algoritmo de otimização TPE (*Tree-Structured Parzen Estimator*) (BERGSTRA et al., 2011).

O algoritmo em suma é uma formalização da otimização Bayesiana baseada em modelo sequencial, que visa aproximar valores de um determinado objetivo com base em medições históricas. De modo a reduzir falhas do processo Gaussiano (RASMUSSEN; WILLIAMS, 2005), que modela diretamente $p(x|y)$, o TPE modela $p(y).p(x|y)$. Neste processo substituem-se as principais distribuições gaussianas de configurações por densidades não paramétricas. Para cada observação das densidades $x^{(1)}, ..., x^{(k)}$, um algoritmo de aprendizado pode produzir sobre um conjunto de configurações existentes. De tal modo que $p(x|y)$, é definidor pela Equação 3.1:

$$p(x|y) = \begin{cases} l(x) & se \quad y < y* \\ g(x) & se \quad y \geq y* \end{cases} \tag{3.1}$$

sendo $l(x)$ a densidade formada pela observação de $x^{(i)}$ tal que a perda correspondente $f(x^{(i)})$ seja menor que $y*$ e $g(x)$ é a densidade formada pelo uso das observações restantes.

Desta forma, os parâmetros das funções de custo de classificação são definidos com base no histórico da soma dos custos, já que a convergência de ambas as funções dependem uma da outra. De modo que, cada parâmetro x com alta probabilidade para $l(x)$ e baixa probabilidade para $g(x)$, seja avaliado de acordo com a Equação 3.2 possibilitando que ao fim de cada iteração, o candidato $x*$ com maior EI (*Expected Improvement*) seja retornado.

$$EI(x) = \frac{l(x)}{g(x)} \tag{3.2}$$

Por fim, após a estimação dos novos valores para as funções de custo reinicia-se a etapa referente a Seção 3.3. Esse processo perdura por 20 iterações para cada combinação de funções classificação/regressão e por fim obter os melhores parâmetros proveniente das configurações atuais de treinamento.

3.5 Avaliação de Resultados

Mediante o exposto, para avaliar os resultados obtidos é necessário utilizar uma métrica comum para todos os modelos, já que as respectivas funções objetivas operam em espaços de valores diferentes. Em vista disso, os modelos são avaliados usando a métrica

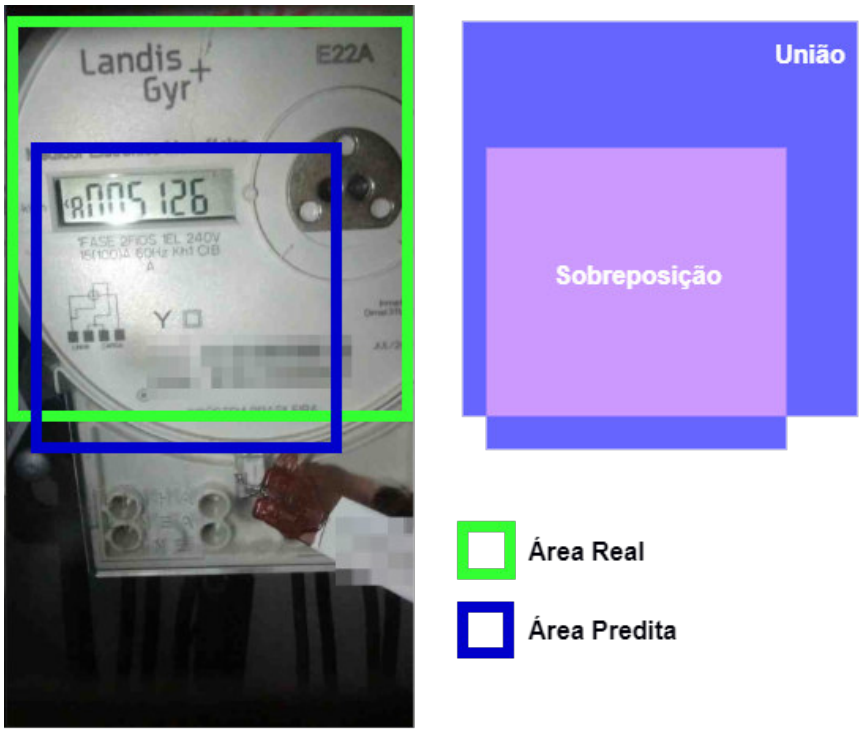
mAP (*mean Average Precision*) em relação a um limiar de regressão IoU (*Intersection over Union*) (HUI, 2018).

3.5.1 IoU - Intersection over Union

A interseção sobre a união nada mais é que a maneira de mensurar a sobreposição entre duas regiões delimitadas, definido pela Equação 3.3, quanto mais próximo de 1, mais preciso será a área predita.

$$IoU = \frac{área\ de\ sobreposição}{área\ da\ união} \tag{3.3}$$

Figura 22 – Elementos de operação da IoU



Fonte: Autor

3.5.2 mAP - mean Average Precision

Métrica comumente utilizada para mensurar a precisão de técnicas de detecção de objeto, o mAP calcula a média da precisão (Equação 3.4), em relação aos valores de *recall* (Equação 3.5). A precisão define o percentual de previsões corretas, o *Recall* define o percentual de previsões corretas do total de casos positivos (HUI, 2018).

$$Precisão = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Positivo} \tag{3.4}$$

$$Recall = \frac{Verdadeiro\ Positivo}{Verdadeiro\ Positivo + Falso\ Negativo} \tag{3.5}$$

Diante disso, o mAP pode ser definido pela Equação 3.6, na qual o total é o somatório da precisão p em função do *recall* r (onde r varia de 0 a 1), dividido pelo total de indivíduos N. Onde para cada acerto deve ter IoU maior ou igual a um determinado percentual preestabelecido, quanto maior o IoU menor será o mAP (HUI, 2018).

$$mAP = \frac{1}{N} \int_0^1 p(r)dr \tag{3.6}$$

Neste capítulo foi apresentado uma metodologia de avaliação das funções de custo. O próximo capítulo apresenta os resultados da aplicação desta metodologia.

4 Resultados e Discussão

Este capítulo apresenta a análise dos resultados obtidos após execução de todo o processo definido anteriormente na Metodologia (Capítulo 3). Os experimentos foram realizados em um computador com processador Intel i7 7ª geração, placa de video Nvidia GeForce GTX 1080 TI, 16Gb de memória RAM. A metodologia foi desenvolvida na linguagem Python[1], versão 3.6.8, utilizando a biblioteca de Tensorflow[2], versão 1.13.0.

Inicialmente, os primeiros testes foram apenas de definição de parâmetros das funções de custo de classificação. Nessa fase o único objetivo foi encontrar os parâmetros que resultavam nas melhores métricas de precisão do algoritmo, já que para cada função de custo o espaço de ponderação é diferente. A biblioteca de otimização *hyperopt* (BERGSTRA; YAMINS; COX, 2013) executou 20 evoluções de otimização para cada combinação de função de classificação e regressão, obtendo os seguintes parâmetros apresentados na Tabela 2.

Tabela 2 – Parâmetros resultantes para funções de custo de classificação.

Cross Entropy	**Focal**		**Correntropy**	
α	α	γ	α	σ
0.96	0.96	5	5	0.92

A função *Cross Entropy*, embora a base tenha um desbalanceamento extremo de classes, o algoritmo de otimização definiu um parâmetro $\alpha = 0.96$, bem próximo de 1. Em contrapartida, a função Focal, decorrente do alto valor de α, também definiu um $\gamma = 5$ alto, aproximando mais a curva de aprendizado dos eixos para instâncias fáceis. Esses parâmetros afetam diretamente na quantidade de épocas necessárias para estabilização do gradiente de aprendizado. Por fim, a função *Correntropy* obteve como melhor ponderador $\alpha = 5$ e alta largura de kernel $\sigma = 0.92$. Uma das causas dos valores estarem bem próximos dos limites do espaço de busca é a dependência direta com as funções de regressão. Se estas funções não obtiverem as classificações mais precisas possíveis a otimização das *bounding boxes* é diretamente afetada.

A Tabela 3 evidencia a eficácia de cada função de classificação em relação à cada função de custo de regressão, ou seja, o quão a precisão de tratamento de erros das funções afetaram na convergência da detecção dos objetos no contexto de medidores de energia elétrica.

Por conseguinte, os resultados da precisão média em relação ao IoU (50%, 75% e 90%) na Tabela 4. Pode ser observado que dependo do nível de precisão desejado a melhor

[1] https://www.python.org/
[2] https://www.tensorflow.org/

Tabela 3 – Resultados de custo de regressão.

	MAE	**MSE**	**Smooth-L1**
Cross Entropy	0,750	0,233	0,478
Focal	**0,627**	**0,175**	0,453
Correntropy	0,628	0,181	**0,440**

combinação entre função de custo de classificação e regressão, para as condições atuais de treinamento, pode ser diferente. Por exemplo:

- IoU = 50%:
  - Cross Entropy.
  - Smooth-L1.
- IoU = 75%:
  - Correntropy.
  - MSE.
- IoU = 90%:
  - Correntropy.
  - Smooth-L1.

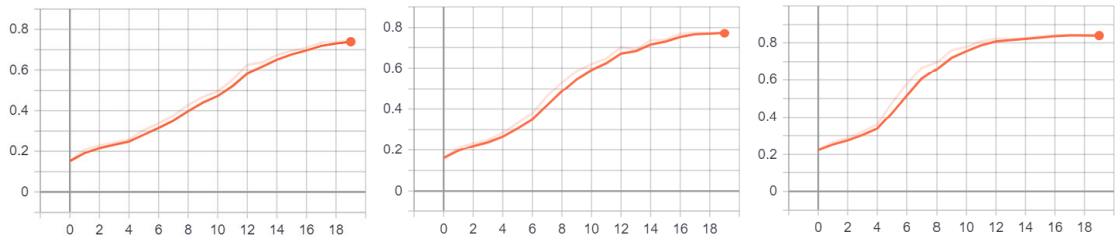
Tabela 4 – Resultados de mAP das respectivas combinação de funções, variando o IoU.

	MAE			**MSE**			**Smooth-L1**		
IoU	**50%**	**75%**	**90%**	**50%**	**75%**	**90%**	**50%**	**75%**	**90%**
Cross Entropy	74,5%	62,1%	6,5%	77,3%	65,4%	5,5%	**84,3%**	72,4%	8,2%
Focal	46,4%	41,4%	7,5%	54,9%	49,1%	8,4%	53,6%	48,5%	8,0%
Correntropy	78,6%	71,0%	10,5%	82,6%	**72,5%**	7,8%	79,0%	71,8%	**10,6%**

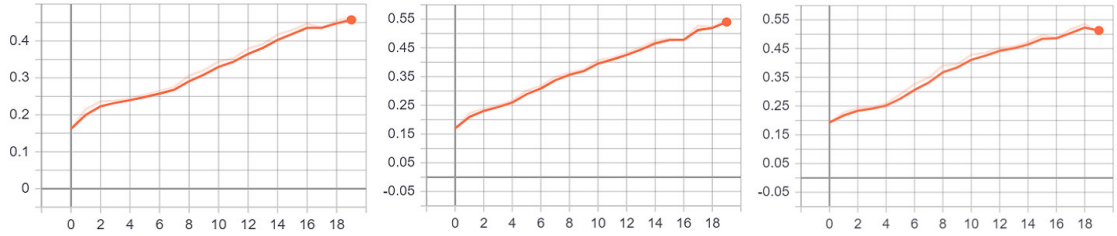
Embora a função Focal tenha demonstrado as menores mAPs, foi a função que mais manteve estabilidade com a variação de IoU com o menor desvio padrão (Tabela 5), por conta da sua característica principal, que leva menos em conta instância de fácil classificação e tende a focar em exemplos mais difíceis tornando o treinamento um pouco mais custoso em relação ao número de épocas.

Tabela 5 – Desvio padrão da mAP em relação a variação do IoU.

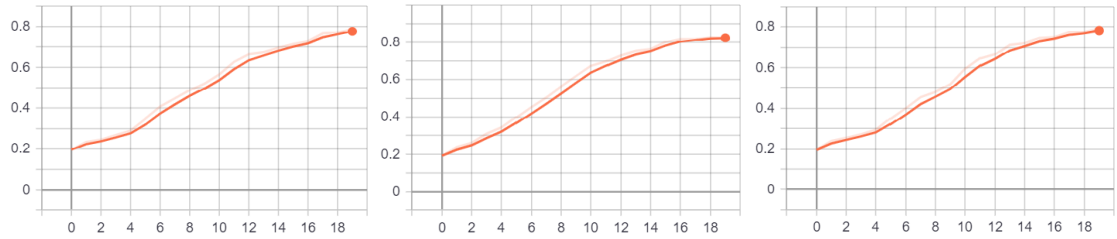
	MAE	**MSE**	**Smooth-L1**
Cross Entropy	0,362	0,385	0,410
Focal	**0,211**	**0,253**	**0,250**
Correntropy	0,373	0,406	0,376



(a) CE : MAE (b) CE : MSE (c) CE : Smooth



(d) FL : MAE (e) FL : MSE (f) FL : Smooth



(g) CR : MAE (h) CR : MSE (i) CR : Smooth

Figura 23 – mAP de treinamento

Na Figura 23, podemos observar que a função Focal, ao fim das 20 épocas, ainda não estabilizou o sua taxa de acerto em relação a todas as funções de regressão. Em contrapartida, a função *Cross Entropy*, em menos de 20 épocas já demonstrou estabilização no gradiente de aprendizado, mesmo não conseguindo manter tanta estabilidade quando o IoU cresce.

Já a função *Correntropy* foi a que obteve os maiores valores de precisão com IoU maior que 90% quando combinadas com as funções de regressão MAE e Smooth-L1. Além de manter uma convergência de aprendizado mais próxima da *Cross Entropy* ainda conseguiu apresentar uma robustez ao grande desbalanceamento de instâncias intra classes e de instâncias tidas como simples. Portanto, a combinação conserva os aspectos positivos das funções de classificação anteriores.

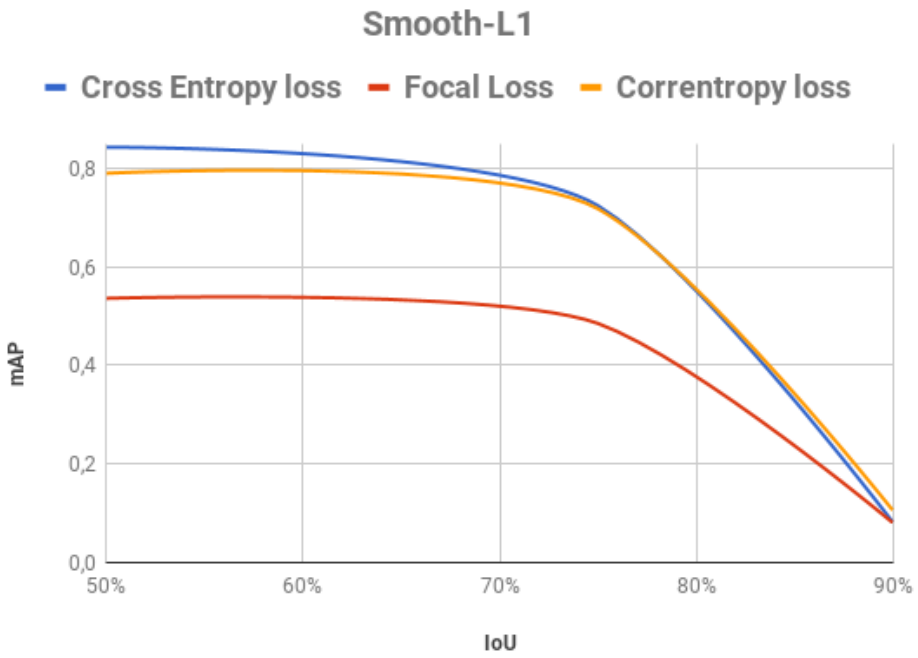
A melhor combinação de funções de custo proposta por Lin et al. (2017b) para o algoritmo Retinanet é Focal/Smooth-L1, com os parâmetros de $\alpha = 0.25$ e $\gamma = 2$. Contudo, para este experimento os melhores resultados, usando a mesma configuração de funções de custo, foram com os parâmetros $\alpha = 0.96$ e $\gamma = 5$ e mesmo assim não alcançou bons resultados de mAP em relação as outras combinações. Na Figura 24 podemos observar que a melhor combinação de funções de custo foi Cross Entropy/Smooth-L1 para o contexto de medidores de energia elétrica. Comumente na literatura os principais valores

de IoU considerados estão entre 50% e 75% e, em ambos, a combinação Focal/Smooth-L1 mostrou-se inferior (Tabela 6).

Tabela 6 – Resultados do mAP para Retinanet.

IoU	**50%**	**75%**
(LIN et al., 2017b)	53,6%	48,5%
Proposto	**84,3%**	**72,4%**

Figura 24 – Variação do mAP em relação ao IoU



Fonte: Autor

5 Conclusões

Este trabalho apresenta uma metodologia para análise do impacto das funções de custo em uma rede neural convolucional de detecção de objetos, no contexto de medidores de energia elétrica. Para conservar a integridade dos dados, os conjuntos de treino e teste foram mantidos em todas as iterações de treinamento.

Para obter os melhores resultados para todas as funções de custo, a metodologia aplica uma otimização Bayesiana baseada em modelo sequencial para inferir os melhores parâmetros possíveis de cada parâmetros dos custos de classificação. Em seguida treina e avalia o modelo para cada combinação possível entre as funções de classificação e regressão. Os melhores resultados de regressão foram obtidos a partir das combinações com a função de custo de classificação Focal, que foi proposta especialmente para os problemas da rede Retinanet (LIN et al., 2017b). Em contraposição, nos resultados de precisão, que em essencialmente é o resultado mais relevante na detecção de objetos, a função focal obteve os piores resultados.

Os melhores resultados de precisão obtidos para IoU 50% foi 84,3% de mAP, com as funções *Cross Entropy* e *Smooth-L1*; para IoU 75%, 72,5% de mAP, com as funções *Correntropy* e MSE; para IoU 90%, 10,6% de mAP, com as funções *Correntropy* e *Smooth-L1*.

Mediante o exposto, observa-se o enorme impacto que a escolha de uma função de custo pode causar nos resultados finais do processo de aprendizado. Já que a função de custo é o avaliador do aprendizado, os demais hiperparâmetros devem ajustar-se a ela. No experimento, os hiperparâmetros foram definidos estaticamente. Isso possibilitou observar que o aprendizado ajustou-se melhor para algumas funções do que para outras.

Portanto, mesmo que a função utilizada resolva todos os problemas provenientes da base de dados e/ou da própria estratégia de aprendizado, ou mesmo aos parâmetros da rede, dentre outros são extremamente determinantes para a escolha da função mais adequada que otimiza os resultados obtidos.

Como trabalhos futuros, propõe-se incluir a validação cruzada na metodologia, aplicar o experimento na base de medidores analógicos. Avaliar os algoritmos que otimizam os pesos das redes neurais, como RMSProp, SGD, dentre outros, já que os mesmos operam diretamente sobre as funções de custo no processo de descida de gradiente durante o treinamento. Otimizar os hiperparametros da rede para demostrar a variação dos mesmos em relação às funções de custo.

Referências

ACADEMY, D. S. *Deep Learning Book*. 2019. <http://www.deeplearningbook.com.br/>. [Online; acessado 2-Novembro-2019]. Citado na página 17.

ARUNAVA. *The Perceptron*. 2018. <https://towardsdatascience.com/the-perceptron-3af34c84838c>. [Online; acessado 30-Outubro-2019]. Citado na página 15.

ARUNAVA. *Rede Neural Convolucional*. 2018. <https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05>. [Online; acessado 11-Novembro-2019]. Citado 2 vezes nas páginas 17 e 18.

BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. Algorithms for hyper-parameter optimization. In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. USA: Curran Associates Inc., 2011. (NIPS'11), p. 2546–2554. ISBN 978-1-61839-599-3. Disponível em: <http://dl.acm.org/citation.cfm?id=2986459.2986743>. Citado na página 34.

BERGSTRA, J.; YAMINS, D.; COX, D. D. *Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms*. 2013. Citado 2 vezes nas páginas 34 e 37.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738. Citado na página 22.

BOYD, S. Convex optimization: From embedded real-time to large-scale distributed. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2011. (KDD '11), p. 1–1. ISBN 978-1-4503-0813-7. Disponível em: <http://doi.acm.org/10.1145/2020408.2020410>. Citado na página 22.

BROWNLEE, J. *Gradient Descent For Machine Learning*. 2016. <https://machinelearningmastery.com/gradient-descent-for-machine-learning/>. [Online; acessado 9-Novembro-2019]. Citado na página 16.

BROWNLEE, J. *Uma introdução suave aos gradientes explosivos em redes neurais*. 2016. <https://machinelearningmastery.com/exploding-gradients-in-neural-networks/>. [Online; acessado 9-Novembro-2019]. Citado na página 16.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2005. (CVPR '05), p. 886–893. ISBN 0-7695-2372-2. Disponível em: <https://doi.org/10.1109/CVPR.2005.177>. Citado na página 19.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>. Citado na página 13.

FENG, Z.; KITTLER, J.; AWAIS, M.; HUBER, P.; WU, X. Wing loss for robust facial landmark localisation with convolutional neural networks. *CoRR*, abs/1711.06753, 2017. Disponível em: <http://arxiv.org/abs/1711.06753>. Citado na página 29.

GANESH, P. *Object Detection : Simplified*. 2019. <https://towardsdatascience.com/object-detection-simplified-e07aa3830954>. [Online; acessado 13-Novembro-2019]. Citado na página 19.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 12.

GROVER, P. *5 Regression Loss Functions All Machine Learners Should Know*. 2018. <https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0>. [Online; acessado 15-Outubro-2019]. Citado 3 vezes nas páginas 27, 28 e 29.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun 2016. Disponível em: <http://dx.doi.org/10.1109/CVPR.2016.90>. Citado 3 vezes nas páginas 18, 19 e 33.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, Elsevier, v. 70, n. 1-3, p. 489–501, 2006. Citado na página 13.

HUI, J. *mAP (mean Average Precision) for Object Detection*. 2018. <https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173>. [Online; acessado 03-Outubro-2019]. Citado 2 vezes nas páginas 35 e 36.

JAY, P. *The intuition behind RetinaNet*. 2018. <https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d>. [Online; acessado 13-Novembro-2019]. Citado na página 20.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2014. Citado na página 33.

LIN, T.-Y.; DOLLAR, P.; GIRSHICK, R.; HE, K.; HARIHARAN, B.; BELONGIE, S. Feature pyramid networks for object detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul 2017. Disponível em: <http://dx.doi.org/10.1109/CVPR.2017.106>. Citado na página 21.

LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLAR, P. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct 2017. Disponível em: <http://dx.doi.org/10.1109/ICCV.2017.324>. Citado 7 vezes nas páginas 21, 23, 25, 33, 39, 40 e 41.

MAHMOUDI, J.; ARJOMAND, M. A.; REZAEI, M.; MOHAMMADI, M. H. Predicting the earthquake magnitude using the multilayer perceptron neural network with two hidden layers. *Civil engineering journal*, v. 2, n. 1, p. 1–12, 2016. Citado na página 12.

RASMUSSEN, C. E.; WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA: The MIT Press, 2005. ISBN 026218253X. Citado na página 34.

REN, L. Y. Z. Correntropy-based robust extreme learning machine for classification. *Neurocomputing*, Elsevier, 2018. Citado 3 vezes nas páginas 12, 13 e 26.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 39, n. 6, p. 1137–1149, Jun 2017. ISSN 2160-9292. Disponível em: <http://dx.doi.org/10.1109/TPAMI.2016.2577031>. Citado na página 19.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 39, n. 6, p. 1137–1149, Jun 2017. ISSN 2160-9292. Disponível em: <http://dx.doi.org/10.1109/TPAMI.2016.2577031>. Citado na página 20.

RICHARD, J. F. C. Divergência de kullback-leibler: uma aplicação à modelagem. 2013. Citado na página 23.

ROSENBLATT, F. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Buffalo, NY, 1961. Citado na página 15.

Mean absolute error. In: SAMMUT, C.; WEBB, G. I. (Ed.). *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. p. 652–652. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_525>. Citado na página 27.

Mean squared error. In: SAMMUT, C.; WEBB, G. I. (Ed.). *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. p. 653–653. ISBN 978-0-387-30164-8. Disponível em: <https://doi.org/10.1007/978-0-387-30164-8_528>. Citado na página 28.

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 7 1948. Disponível em: <https://ieeexplore.ieee.org/document/6773024/>. Citado na página 23.

SINGH, A.; POKHAREL, R.; PRINCIPE, J. The c-loss function for pattern classification. *Pattern Recognition*, v. 47, n. 1, p. 441 – 453, 2014. ISSN 0031-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0031320313003154>. Citado 3 vezes nas páginas 13, 25 e 27.

TORRE, J. de L.; PUIG, D.; VALLS, A. Weighted kappa loss function for multi-class classification of ordinal data in deep learning. *Pattern Recognition Letters*, Elsevier, v. 105, p. 144–154, 2018. Citado na página 14.