Identificação do perfil de clientes utilizando Redes Neurais Convolucionais

Victor Ribeiro de Azevedo
Programa de Pós-graduação em Engenharia Eletrônica
UERJ, Brasil
Email: v.azevedo9@gmail.com

Nadia Nedjah
Programa de Pós-graduação em Engenharia Eletrônica
UERJ, Brasil
Email: nadia@eng.uerj.br

Luiza de Macedo Mourelle
Programa de Pós-graduação em Engenharia Eletrônica
UERJ, Brasil
Email:ldmm@eng.uerj.br

***Resumo*—Neste trabalho são utilizadas as técnicas de redes neurais convolucionais e aprendizagem profunda a fim de prever o interesse de usuários de redes sociais em determinadas categorias de produtos. O objetivo consiste em realizar a classificação de imagens de interesses de um certo tipo de usuário de redes sociais. A classificação de imagens permite segmentar usuários de redes sociais como potenciais consumidores de determinados tipos de produtos. Para isto foi realizada a comparação do desempenho dos seguintes algoritmos de taxa de aprendizagem adaptativa de redes neurais artificiais: descida do gradiente estocástico, descida de encosta adaptativa, estimativa de momento adaptativo e suas variações baseado na norma infinita e na raiz quadrada média dos gradientes. A comparação dos algoritmos de treinamento mostra que o algoritmo de estimativa de momento adaptativo é o mais adequado para prever o interesse e o perfil do usuário. A classificação de imagens em 17 subcategorias alcançou uma precisão de classificação de aproximadamente 99%.**

***Keywords*—Redes Neurais Convolucionais, Aprendizagem profunda, Classificação imagens de redes sociais, Identificação do perfil de clientes**

I. Introdução

Com o grande volume de informações digitais na forma de imagens e vídeos nas redes sociais, torna-se necessário o desenvolvimento de novas técnicas para extrair informações relevantes dos usuários. Milhares de imagens estão sendo publicadas em redes sociais diariamente, com isto, a classificação de imagens de usuários de redes sociais, se demonstra um processo relevante para empresas. Com a segmentação dos clientes é possível definir potenciais consumidores de seus produtos, ou ainda, criar propagandas direcionadas para públicos distintos.

O *Instagram* é uma das redes sociais mais populares e possui mais de 1 bilhão de usuários ativos mensais conforme estimado por especialistas em [2] no ano de 2018. Essa quantidade é mais do que o triplo dos usuários ativos mensais do Twitter, não obstante, está atrás dos usuários do *WhatsApp* e do *Facebook Messenger*. Todos os dias, os usuários do Instagram publicam em média mais de 100 milhões de fotos conforme relatado em [2].

Vale ressaltar que 72% usuários do *Instagram* dizem que compraram um produto que viram no aplicativo conforme mencionado em [9]. Isto demonstra que segmentar os usuários por interesse é uma estratégia relevante de vendas.

Acredita-se que este trabalho contribui para comunidade científica, no ramo de visão computacional e classificação de imagens, com seu caráter exploratório e experimental na análise do desempenho de redes neurais artificiais.

Este trabalho demonstra a utilização de Redes Neurais Convolucionais (*Convolutional Neural Network* - CNN) com a utilização de variações do algoritmo de treinamento para a classificação de imagens nas seguintes categorias: animais, eletrônicos, jogos, veículos e vestuário. Para melhorar o desempenho do sistema de aprendizagem profunda foram realizados experimentos variando a estrutura do modelo em relação aos algoritmos de treinamento de aprendizagem profunda e a inicialização de hiperparâmetros como número de épocas e taxa de aprendizado. Foi realizada a comparação do desempenho dos seguintes algoritmos de treinamento : descida de encosta estocástica (*stochastic gradient descent* - SGD), gradiente adaptativo (adaptive gradient - *AdaGrad*), gradiente adaptativo baseado na norma infinita (*adaptive stochastic gradient descent* - Adamax), algoritmo de propagação do quadrado médio da raiz (*root mean square prop* - RMSprop) e o algoritmo de estimativa de momento adaptativo (*adaptive moment estimation* - Adam).

O restante deste trabalho está dividido em 5 seções. Primeiro, na Seção II, apresentamos os trabalhos relacionados. Depois, na Seção III, introduzimos os conceitos relacionados as redes neurais convolucionais. Em seguida, na Seção IV, abordamos a solução proposta e a metodologia técnica utilizada. Em sequência, na Seção V, comentamos e analisamos os resultados obtidos. Finalmente, na Seção VI, discutimos as conclusões deste trabalho e apontamos algumas direções para trabalhos futuros.

II. Trabalhos relacionados

Em [12] é feita a inferência da personalidade de usuários da rede *Flickr* a partir de imagens de suas galerias. Foi investigada a interação entre as postagens que as pessoas deixam nas redes sociais e traços de suas personalidade. Os experimentos realizados com 60.000 imagens adicionadas aos favoritos por 300 usuários permitiram extrair características e segmentar os usuários.

Em [5] é realizada a inferência do interesse de usuários da rede social *Pinterest* baseado na classificação de textos e

imagens. O trabalho visa a recomendação de produtos de determinadas marcas baseado na classificação de imagens postadas pelos usuários. Os autores relatam que um dos desafios do projeto é a interpretação do interesse do usuário quando a imagem ou comentário postado na rede social contém o produto a ser classificado, porém o usuário refere-se a outro assunto. Diante deste fato, foi realizada a comparação dos resultados utilizando um espaço multi-modal, combinando textos e imagens, e uni-modal, com textos ou imagens, a fim de alcançar uma melhor acurácia no resultado. Os autores utilizam CNNs para classificação das imagens, da técnica de *Bag of Words* (BoW) para representação textual e da técnica *Bag of Visual Words* (BoVW) para representação da imagem como um mapa de características. Para obtenção das imagens e dos textos como fonte de dados dos usuários foi realizado um programa que percorreu 650 usuários obtendo 443.000 conjuntos de textos e imagens denominados *pins* no *Pinterest*. Em seguida os dados foram divididos em treinamento, validação e teste para realização de onze experimentos utilizando diferentes combinações dos algoritmos supracitados.

Em [4] é feita a seleção de um conjunto de imagens da rede social *Flickr* para detecção de objetos.É feita a clusterização de regiões das imagens utilizando o algoritmo de Propagação por afinidade (*Affinity propagation*). Os autores utilizam aprendizagem supervisionada na classificação de objetos com modelos de reconhecimento de imagens.

Em [3] são realizadas classificações de objetos de redes sociais em 101 categorias. As bases de dados utilizadas contem imagens da rede social *Flickr*.

III. Redes Neurais Convolucionais

Redes neurais convolucionais são modelos de aprendizagem profunda geralmente utilizados para aplicações de reconhecimento e classificação de padrões em imagens e vídeos. Uma CNN é geralmente formada por: uma camada de entrada que recebe as imagens, uma série de camadas que realizam operações de tratamento de imagens com mapeamento de características e uma última etapa que contém uma rede neural de classificação, que recebe o mapeamento de características e fornece como saída o resultado da classificação.

As CNNs são compostas por neurônios que possuem pesos e limiares (*bias*) que necessitam ser treinados. Cada neurônio recebe entradas, aplica-se o produto escalar das entradas e pesos além de uma função não-linear (função de ativação). Uma CNN assume que todas as entradas são imagens, o que permite definir propriedades na arquitetura. Redes neurais tradicionais não são escaláveis para imagens, uma vez que produzem um número muito alto de pesos a serem treinados.

Uma CNN é composta por uma sequência de camadas. Além da camada de entrada, que recebe as imagens, exitem três camadas principais: camada convolucional, camada de *pooling* (agrupamento) e camada completamente conectada. Além disso, após uma camada de convolução é comum uma camada de ativação. Essas camadas, quando colocadas em sequência, formam uma arquitetura de uma CNN como ilustrado na Figura 1. A camada totalmente conectada (*fully connected layer*) é uma rede neural comum que recebe a saída dos mapas de características e realiza a classificação final.

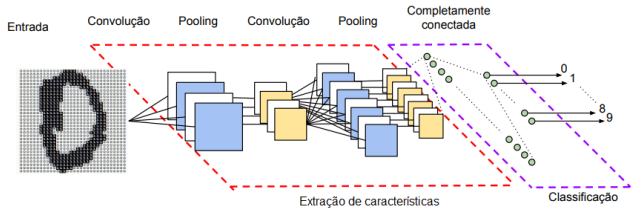

Figura 1. Arquitetura de uma Rede Neural Convolucional

Em [1] são citadas as camadas de agregação (*pooling layer*) como responsáveis por reduzirem a dimensionalidade das matrizes e a complexidade na Rede Neural. Os autores citam os diferentes tipos de camadas agregação e suas vantagens. Os tipos de camadas citados são: *Max pooling, Average pooling, Deformation pooling, Spatial pyramid pooling, Scale dependent pooling*.

A camada de convolução é composta por um conjunto de filtros que são matrizes de valores reais que podem ser interpretados como pesos capazes de aprender de acordo com um treinamento. Esses filtros auxiliam na convolução com os dados de entradas para obter um mapa de características. Estes mapas indicam regiões na qual características específicas em relação ao filtro, são encontradas na entrada. Os valores dos filtros se alteram ao longo do treinamento (assim como os pesos de uma rede neural tradicional) fazendo com que a rede aprenda a identificar regiões significantes de extração de características do conjunto de dados como na Figura 2.

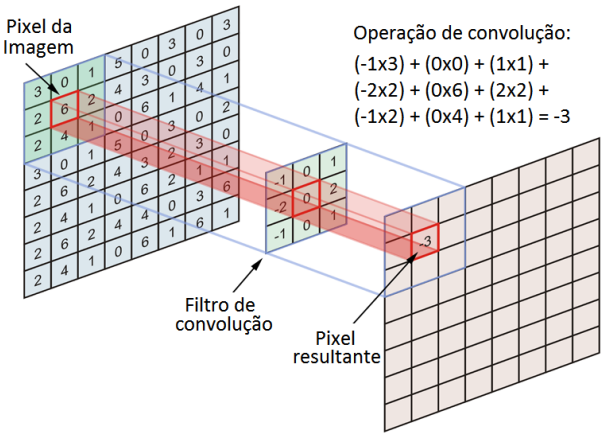

Figura 2. Ilustração do processo de convolução

Nas matrizes de resultados da convolução são aplicadas funções de ativação. A função mais utilizada é a *Rectifier Activation Function* (Função de retificação linear) - ReLU , que é simplesmente aplicar a função de maximização em cada elemento do resultado da convolução. A técnica de *pooling* é utilizada após a camada de convolução com objetivo de reduzir o tamanho espacial das matrizes resultantes da convolução.

A. Algoritmos de taxa de aprendizagem adaptativa

A taxa de aprendizado é uma constante positiva, que corresponde à velocidade do aprendizado. Nos algoritmos de taxa de aprendizagem adaptativa o valor da taxa de aprendizagem é modificado por um termo definido em cada algoritmo. Os principais algoritmos de aprendizagem adaptativos são: AdaGrad, RMSProp e Adam. Além destes, existem outros como Adamax, AdaDelta e Nadam.

IV. Metodologia técnica

Este trabalho trata de uma nova abordagem ao procedimento de classificar imagens e tem como objetivo avaliar a performance do treinamento de CNNs com algoritmos alternativos em comparação com a descida de encosta. Nesta seção são apresentados os procedimentos utilizados para o desenvolvimento do modelo para o reconhecimento de imagens de redes sociais em 5 categorias definidas como animais, eletrônicos, jogos, veículos e vestuário.

Foram utilizados conjuntos de imagens de produtos, uma rede neural convolucional com as seguintes variações dos algoritmos: SGD, AdaGrad, Adamax, RMSProp e Adam. Para análise dos resultados foram apresentados gráficos com as métricas : erro, acurácia, sensibilidade e a medida F1. Por fim foram realizadas predições de cada uma das categorias animais, eletrônicos, jogos, veículos e vestuário definidas para uma amostra de imagens de teste.

Para o desenvolvimento do modelo de treinamento foi criado um ambiente virtual com o contêiner *docker*. Um contêiner *docker* é um ambiente encapsulado padronizado que executa aplicativos por meio da virtualização em nível do sistema operacional. Uma imagem do *docker* é um arquivo, composto de várias camadas, usado para executar código em um contêiner do *docker*. Foi utilizada a imagem [10] de um container *docker* com as seguintes ferramentas: *Jupyter*, *Matplotlib*, *Pandas*, *Tensorflow*, *Keras* e *OpenCV*. Vale salientar que neste trabalho foi utilizado o *Tensorflow* e o *Keras* com a linguagem *Python* para realização do treinamento.

O *Tensorflow* é uma biblioteca de software de código aberto para a inteligência de máquina criado pelo Google em 2015. Com o *Tensorflow* é possível definir modelos de aprendizado de máquina, treiná-los com dados e exportá-los.

O *Tensorflow* opera com tensores que são vetores multidimensionais que percorrem pelos nós de um grafo. Este grafo é composto pelos seguintes elementos:

- Um conjunto de objetos *tf.Operation*, que representam as unidades com as operações
- Um conjunto de *tf.Tensor*, que representam as unidades com os dados
- Uma sessão *tf.Session* que encapsula o ambiente onde as operações do grafo são executadas e os tensors são avaliados.

Ao final dos experimentos, o desempenho de cada algoritmo foi analisado quanto ao erro (*error*), acurácia (*accuracy*), sensibilidade (*recall*) e a medida F1 (*FMeasure*). A definição de cada um destes parâmetros é apresentada abaixo:

- Acurácia: Taxa de acertos que o modelo obteve.
- Precisão: Taxa de acertos positivos dentre os classificados como positivos.
- Especificidade: Taxa de acertos negativos dentre os que são negativos.
- Sensibilidade: Taxa de acertos positivos dentre os que são positivos.

Para medição destas métricas de desempenho foi utilizada a biblioteca de código aberto *sklearn*. Segundo a documentação do *sklearn* é possível definir as métricas, variando de um mínimo de 0 a um máximo de 1 de forma adimensional, conforme Equação 1:

$$\begin{array}{ll} precision = & \frac{TP}{TP+FP} \\ recall = & \frac{TP}{TP+FN} \\ F1 = & \frac{2 \times precision}{precision+recall} \\ accuracy = & \frac{TP+TN}{TP+FN+TN+FP} \\ specificity = & \frac{TN}{TN+FP}, \end{array} \quad (1)$$

onde TP é o número de verdadeiros positivos, FP são os falsos positivos, TN é o número de verdadeiros negativos e FN são os falsos negativos.

Como descrito anteriormente, o modelo foi testado com 5 variações de algoritmos. Os resultados de cada abordagem foram então comparados e ordenados de acordo com sua performance e desempenho em relação a acurácia e erro.

A. Obtenção dos dados

A seleção das imagens para treino foi baseada nos *datasets* dos bancos de imagens do *Kaggle*, *pyimagesearch* e do *dataset* [6]. Foram utilizadas cerca de 1.500 imagens de 5 categorias dividas em 17 subcategorias conforme descrito:

- Animais domésticos: Cachorros, gatos e pássaros (264 imagens);
- Dispositivos Eletrônicos: Celulares, Notebooks e Televisões (366 imagens);
- Jogos: Consoles e cenas de jogos (188 imagens);
- Veículos: Carros, motos, bicicletas e avião (386 imagens);
- Vestuário: Calças, Vestidos, Blusas, Sapatos e Tênis (298 imagens);

Para os animais foi utilizado o *dataset* [8], para os veículos foi utilizado o [7], para o vestuário foi utilizado o [11], para eletrônicos foi utilizado o [6] e as demais imagens foram obitdas de redes sociais de usuários. As imagens foram organizadas em pastas para realização de aprendizado supervisionado, sendo uma pasta para cada uma das categorias.

B. Desenvolvimento do modelo

As imagens utilizadas no treinamento passaram por técnicas de processamento visando aumentar a performance do modelo classificador e diminuir o esforço computacional necessário para processá-las.

O conjunto de imagens foi divido em 80% para treinamento e 20% para teste. As imagens foram redimensionadas para um padrão de 96px de largura e 96px de altura. A intensidade dos *pixels* foi normalizada dividindo seus valores escalares por 255

de forma para variarem num intervalo de 0 a 1. Com isto as matrizes das imagens foram representadas com uma dimensão de 96x96x3 com 3 canais de cores conforme a escala de cores RGB (*Red Green Blue*).

A arquitetura da CNN que foi utilizada foi introduzida em [13] e é uma variante menor da rede VGGNet, desenvolvida pelo VGG *Visual Geometry Group* da Universidade de *Oxford*.

A arquitetura da rede foi definida com 7 camadas de convolução 2D, com redução da dimensionalidade das matrizes utilizando a função *max pooling* com dimensão 2x2 e 3x3, com filtros (*kernels*) de 3x3 e funções de ativação *relu*. A camada final totalmente conectada é criada utilizando a função *softmax* para classificação. São utilizadas 6 camadas com a sequencia de operações de convolução, função de ativação *relu* e *pooling* e por fim uma camada com totalmente conectada com a função *softmax* para classificação.

Foi adicionada na rede camadas de *Dropout*, que trata de uma técnica utilizada para controlar o problema de *overfitting*, aumento do erro devido a excesso de treinamento. O *overfitting* ocorre quando a RNA memoriza os padrões de treinamento em vez de aprendê-los e a rede não possui boa capacidade de generalização. A camada de *Dropout* zera uma certa quantidade de ativações da camada baseando-se em uma tomada de decisão probabilística, agindo mediante ativação por um limiar pré-definido. A rede deve ser capaz de classificar dados corretamente mesmo sem contar com algumas de suas ativações em cada iteração. Esta técnica se torna importante para evitar o processo de *overfitting*. A rede então processa as imagens nas seguintes etapas:

- Obtenção das características: Nesta primeira fase a imagem é processada com os filtros convolucionais para detectar o grau de pertinência da imagem com as características buscadas.
- Posicionamento das características: Nesta segunda fase, os mapas de características definem o que existe na imagem e aonde está cada característica.

Em resumo, a primeira fase se preocupa em encontrar o que existe na imagem e a segunda em definir aonde estão essas características na imagem.

V. Resultados de desempenho

Com o conjunto de dados preparados as rotinas de treinamento puderam ser estabelecidas sendo executadas no container *docker* utilizando a biblioteca *keras* e com o *python*.

A. Treinamento via algoritmo SGD

No algoritmo da descida de encosta estocástica utiliza-se uma amostra de treinamento para atualizar os parâmetros em uma iteração específica. Já no algoritmo de de descida de encosta (*Gradient Descent* - GD) é necessário percorrer todas as amostras em seu conjunto de treinamento para realizar a atualização dos parâmetros em uma iteração específica.

Com o número de amostras de treinamento grande o uso do GD pode demorar muito. Por outro lado, usar o SGD será mais rápido porque a melhora do desempenho ocorre a partir da primeira amostra. O SGD geralmente converge muito mais rápido comparado ao GD , não obstante, a função de erro no GD é melhor minimizada. Foram realizados treinamentos com o algoritmo SGD utilizando 20 épocas.

Na Figura 3 é possível observar o decaimento do erro ao longo das épocas para o treinamento e teste. Nesta figura, o erro alcança o valor de 0,45 para treinamento e 0,46 para teste. A curva de erro de teste tem algumas instabilidades pontuais, mas as duas curvas de treinamento e teste mostram claros decaimento de erro e aumento da precisão.

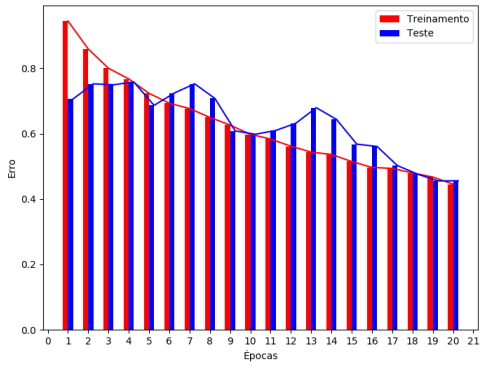

Figura 3. Decaimento do erro durante o treinamento e teste utilizando o algoritmo *SGD*

Na Figura 4 é possível visualizar o crescimento da precisão do treinamento e de validação, atingindo um desempenho de 80,80% para treinamento e 82,27% para teste.

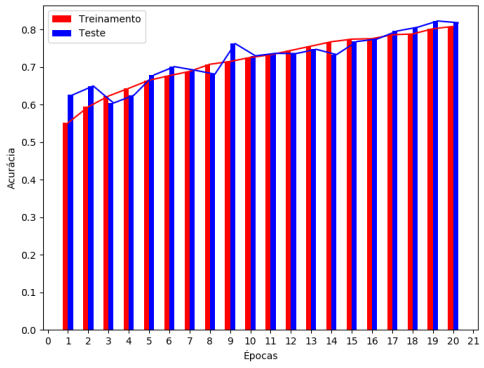

Figura 4. Crescimento da acurácia durante o treinamento e teste utilizando o algoritmo *SGD*

Na Figura 5 é possível observar o aumento da taxa de acerto com a visualização do aumento estável do F1, da *precision* e *recall* mostrando que os resultados relevantes poderiam ser corretamente classificados pelo algoritmo.

B. Treinamento via algoritmo AdaGrad

O algoritmo *AdaGrad* tem o efeito de reduzir a taxa de aprendizado dos pesos que recebem gradientes altos

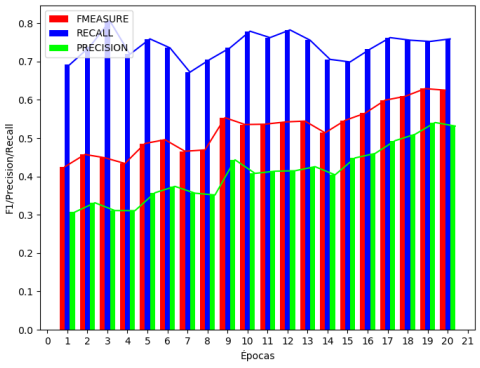

Figura 5. Curvas das métricas: FMeasure, Precision, Recall utilizando o algoritmo *SGD*

e ao mesmo tempo aumentar a taxa de aprendizado nas atualizações. É um método onde a taxa de aprendizado é normalizada pela raiz da soma dos gradientes de cada parâmetro ao quadrado. Para este algoritmo são apresentados os resultados de treinamento utilizando 15 épocas e taxa de aprendizado inicial de 0,01.

Na Figura 6 é possível observar o comportamento do erro de decaimento ao longo das épocas tanto para treinamento quanto para teste. Nesta figura é mostrado o erro decaindo para um mínimo de 0,24 para treinamento e 0,26 para teste. A curva de erro de teste tem algumas instabilidades pontuais como o SGD, mas ambas as curvas de treinamento e teste têm boa estabilidade relativa geral na queda de erros e no aumento da precisão.

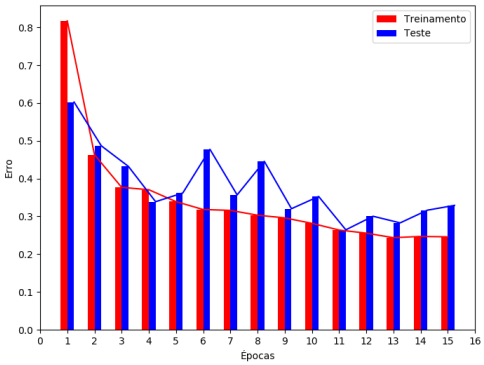

Figura 6. Decaimento do erro durante o treinamento e teste utilizando o algoritmo *AdaGrad*

Na Figura 7 é possível observar um aumento instável da métrica F1, *precision* e *recall*.

Na Figura 8 é possível visualizar o crescimento da precisão do treinamento e a precisão da validação atingindo um desempenho de 90.21% para treinamento e 89.5% para teste demostrando melhores e resultados mais estáveis que o algoritmo SGD .

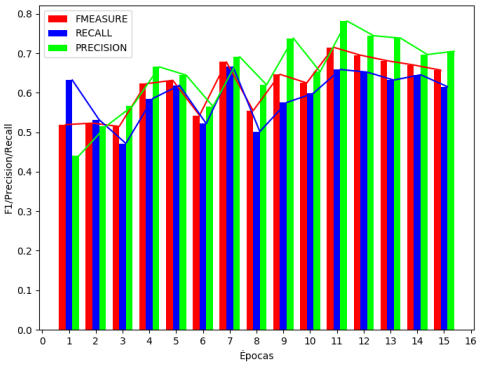

Figura 7. Curvas das métricas: FMeasure, Precision, Recall utilizando o algoritmo *AdaGrad*

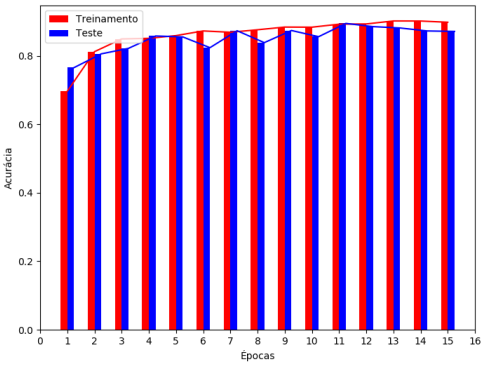

Figura 8. Crescimento da acurácia durante o treinamento e teste utilizando o algoritmo *AdaGrad*

C. Treinamento via algoritmo RMSprop

No algoritmo *RMSprop* é realizado o cálculo das médias da magnitude dos gradientes para cada parâmetro e elas são utilizadas para modificar a taxa de aprendizado individualmente antes de aplicar os gradientes. Este algoritmo modifica o *AdaGrad* para atingir um melhor desempenho. Foi realizado o treinamento utilizando o *RMSprop* com 20 épocas e taxa de aprendizado inicial de 0,001.

Na Figura 9 é possível verificar o erro decaindo ao mínimo de 0,20 para treinamento e 0,37 para teste. Ambas as curvas de treinamento e teste apresentam boa estabilidade relativa geral no aumento da acurácia conforme observado.

Na Figura 10 é possível observar um aumento instável das métricas F1, *precision* e *recall*. O aumento dessas métricas ao longo de épocas reflete o aumento da taxa de acertos.

Na Figura 11 é possível visualizar o crescimento da precisão do treinamento e a acurácia da validação, atingindo um desempenho de 92,19% no treinamento e 89.16% no teste.

D. Treinamento via algoritmo Adam

O algoritmo de treinamento de estimativa de momento adaptativo é um método derivado do *RMSprop* que ajusta o

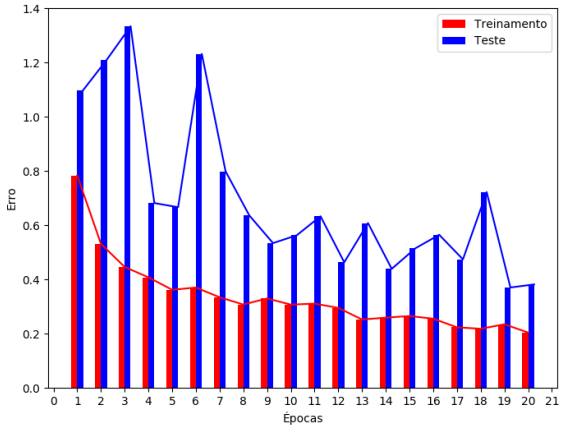

Figura 9. Decaimento do erro durante o treinamento e teste utilizando o algoritmo *RMSprop*

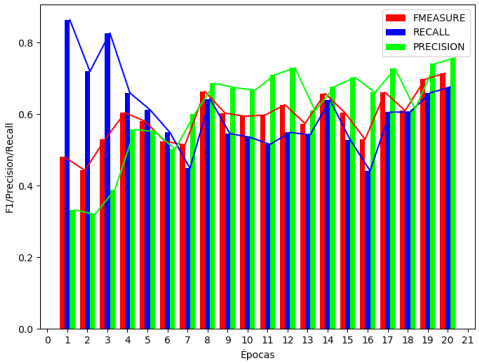

Figura 10. Curvas das métricas: FMeasure, Precision, Recall utilizando o algoritmo *RMSprop*

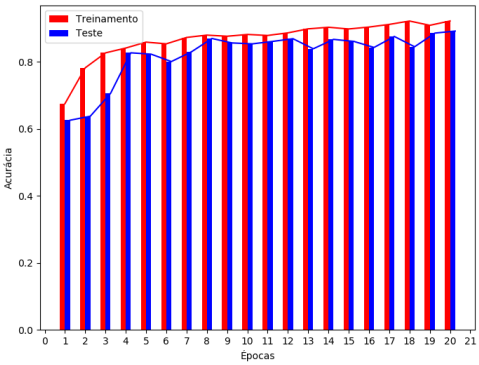

Figura 11. Crescimento da acurácia durante o treinamento e teste utilizando o algoritmo *RMSprop*

método *Adagrad* para que a taxa de aprendizado não diminua agressivamente. A contribuição do método Adam é adicionar um momento na atualização e suavizar os ruídos do gradiente antes de fazer essa operação. Ele herda do *RMSprop* a adição de uma taxa de decaimento na soma dos gradientes de cada parâmetro enquanto a taxa de aprendizado é reduzida a cada passo. São apresentados os resultados do algoritmo utilizando 50 épocas.

A Figura 12 mostra o erro decaindo a um mínimo de 0,12 para treinamento e a 0,23 para teste mostrando o bom desempenho deste algoritmo. A curva de erro de teste tem alguma ocorrência de instabilidades com picos provavelmente relacionados a dados aleatórios escolhidos na etapa de teste.

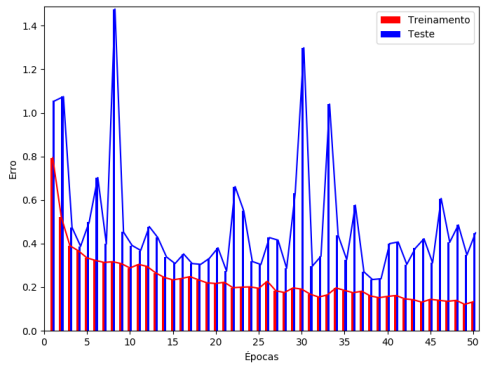

Figura 12. Decaimento do erro durante o treinamento e teste utilizando o algoritmo *Adam*

Na Figura 13 é possível visualizar o crescimento da precisão do treinamento e da validação atingindo desempenho de 95,36% no processo de treinamento. É possível verificar ainda nesta figura, um desempenho de 92,12% para teste demostrando uma alta performance em ambas as etapas. Ambas as curvas de treinamento e teste apresentam boa estabilidade relativa geral no aumento da acurácia.

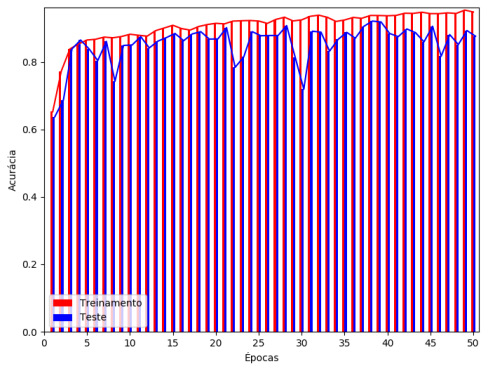

Figura 13. Crescimento da acurácia durante o treinamento e teste utilizando o algoritmo *Adam*

Na Figura 14 é possível observar alguns vales no aumento das métricas F1, *precision* e *recall*. O aumento dessas métricas ao longo de épocas reflete o aumento da taxa de acertos.

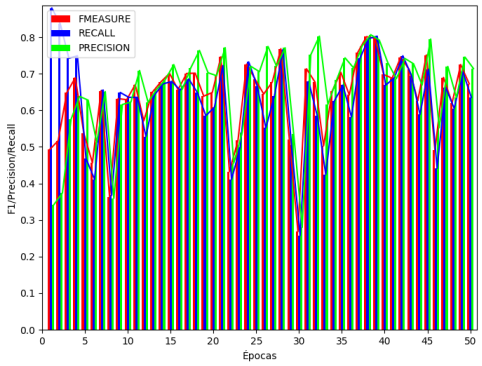

Figura 14. Curvas das métricas: FMeasure, Precision, Recall utilizando o algoritmo *Adam*

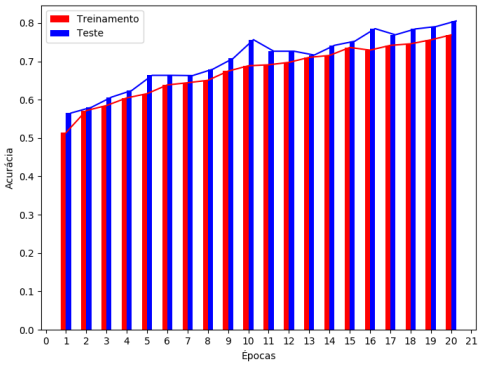

Figura 16. Crescimento da acurácia durante o treinamento e teste utilizando o algoritmo *Adamax*

E. Treinamento via algoritmo Adamax

O Adamax é um algoritmo variante do Adam onde o momento de segunda ordem é substituído pelo momento de ordem infinita. A seguir são apresentados os resultados do algoritmo Adamax utilizando 20 épocas.

A Figura 15 mostra o erro decaindo para 0,23 para treinamento e 0,36 para teste.

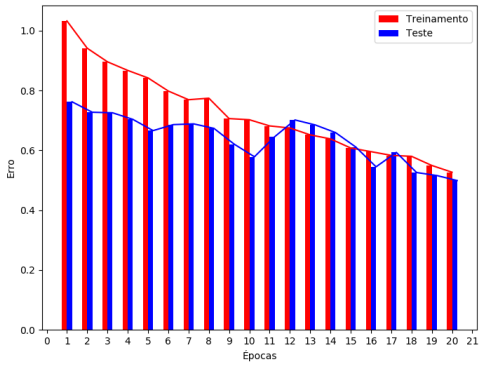

Figura 15. Decaimento do erro durante o treinamento e teste utilizando o algoritmo *Adamax*

Na Figura 16 é possível visualizar o crescimento da acurácia do treinamento alcançando valor máximo de 90,27%. É possível visualizar o crescimento da acurácia de teste alcançando valor máximo de 86,15%.

Na Figura 17 é possível observar o aumento da taxa de acerto com a visualização do aumento estável das métricas F1, da *precision* e *recall*. Todas as curvas de erro, *precisão*, *F1*, *precision* e recuperação são muito estáveis para este algoritmo, conforme observado na Figura 17.

A Tabela I apresenta a acurácia máxima (Acc. max.) e o erro mínimo (Erro min.) para os algoritmos utilizados no treinamento. Além disto são apresentados a quantidade de épocas e taxa de aprendizado (Tx. aprend.) para cada procedimento. Nesta tabela é mostrada que a pior acurácia foi de 80,80% para

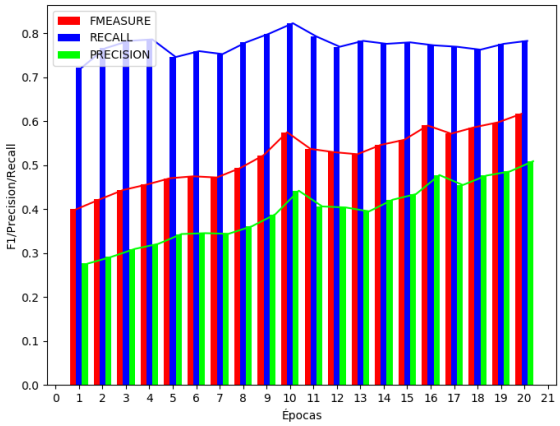

Figura 17. Curvas das métricas: FMeasure, Precision, Recall utilizando o algoritmo *Adamax*

o algoritmo SGD e as duas maiores acurácias foram de 92,19% e 95,36% do algoritmo RMSProp e Adam respectivamente. Os menores erros observados para o treino foi de 0,20 para o RMSProp e de 0,12 para o algoritmo Adam. Com a Tabela I é possível verificar a diferença de desempenho dos algoritmos na etapa de teste observando os valores máximos e mínimos de acurácia e erro respectivamente.

Tabela I
MÉTRICAS DE DESEMPENHO DE TREINAMENTO

Algoritmo	**Épocas**	**Tx. aprend.**	**Acc. max. (%)**	**Erro min.**
SGD	20	0,001	80,80	0,45
AdaGrad	15	0,01	90,21	0,24
Adamax	20	0,002	90,27	0,23
RMSProp	20	0,001	92,19	0,20
Adam	50	0,001	95,36	0,12

A Tabela II apresenta a acurácia máxima (Acc. max.), o erro mínimo (Erro min.) e a taxa de aprendizado (Tx. aprend.) para os algoritmos utilizados no teste bem como a quantidade

de épocas utilizadas. Nesta tabela a pior acurácia foi de 82,27% para o algoritmo SGD e as duas maiores acurácias foram de 89,50% e 92,10% do algoritmo AdaGrad e Adam respectivamente. Os menores erros observados para o teste foi de 0,26 para o AdaGrad e de 0,23 para o algoritmo Adam. Com a Tabela II é possível verificar o desempenho dos algoritmos na etapa de teste observando os valores máximos e mínimos de acurácia e erro respectivamente.

Tabela II
MÉTRICAS DE DESEMPENHO DE TESTE

Algoritmo	Épocas	Tx. aprend.	Acur. max. (%)	Erro min.
SGD	20	0,001	82,27	0,46
Adamax	20	0,002	86,15	0,36
RMSProp	20	0,001	89,16	0,37
AdaGrad	15	0,01	89,50	0,26
Adam	50	0,001	92,10	0,23

Na Tabela III são apresentados os resultados das probabilidades de pertencer a cada categoria nos testes utilizando o algoritmo Adam com o melhor desempenho de treino para imagens das 5 categorias. Vale ressaltar que estes resultados da Tabela III foram obtidos utilizando amostras aleatórias das imagens pertencentes aos dados de testes que não foram utilizados pelos algoritmo no treinamento. Pode-se observar valores superiores a 99% de desempenho demonstrando boa capacidade de generalização.

Vale ressaltar que o desempenho para classificar as categorias cadeira, jardim e torre apresentaram probabilidades máximas de 66,88%, 0% e 38,42% respectivamente demonstrando que essas classes não foram utilizadas no treino.

Diversas imagens das categorias treinadas apresentaram desempenhos superiores a 98% na classificação indicando boa capacidade de generalização.

Tabela III
RESULTADO DA CLASSIFICAÇÃO DE IMAGENS NAS 5 CATEGORIAS

Imagem	Anim.	Eletr.	Jog.	Veíc.	Vest.
Cachorro	99,77	0,00	0,01	0,09	0,00
Gato	99,37	0,01	0,05	0,64	0,07
Pássaro	99,22	0,05	0,14	0,16	0,07
Notebook	0,12	99,59	0,00	0,08	1,87
TV	0,04	82,62	0,03	66,50	0,14
Cel.	0,00	99,98	0,00	0,01	0,01
Ps4	0,01	3,07	95,84	0,01	10,32
Carro	0,00	0,05	7,16	96,92	0,21
Moto	0,00	0,47	0,80	97,53	3,47
Bicicleta	0,00	0,06	0,08	98,63	4,04
Avião	0,02	0,02	0,05	98,78	1,31
Sapato	0,23	1,33	0,21	55,13	95,92
Vestido	1,25	0,00	0,00	0,19	99,70
Calça	12,02	2,34	0,28	0,31	85,49
Cadeira	0,83	3,17	0,04	6,29	66,68
Jardim	0,00	0,00	0,00	0,00	0,00
Torre	0,01	32,76	38,42	1,73	1,78

VI. CONCLUSÃO

Neste trabalho foram utilizadas redes neurais convolucionais e aprendizagem profunda como técnicas de inteligência computacional para predição do interesse de usuários de redes sociais em determinadas categorias de produtos utilizando a classificação de imagens. Esta análise é válida como fonte de conhecimento na segmentação de consumidores por interesse. A comparação dos algoritmos de treinamento SGD, AdaGrad, Adamax, RMSProp e Adam permitiram verificar que o algoritmo de treinamento *Adam* obteve acurácias e performance superiores aos demais.

Algoritmos de treinamento e otimização são uma parte crucial da rede neural. Entender seu desempenho pode ajudar a escolher a melhor opção para os sistemas. O conhecimento de como os hiperparâmetros podem influenciar o desempenho da CNN é muito importante ao treinar uma rede. A comparação dos algoritmos de treinamento SGD, AdaGrad, Adamax, RMSProp e Adam permitiu verificar que o algoritmo de treinamento Adam obteve acurácia e desempenhos superiores aos demais e boa estabilidade.

Para trabalhos futuros, a sugestão é variar as categorias das imagens forma a obter classificações com maior capacidade de generalização.

Após a finalização de todos os experimentos, ao classificar as imagens das categorias definidas foi possível obter desempenhos satisfatórios de classificação.

REFERÊNCIAS

[1] M. Pandey A. R. Pathaka and S. Rautaraya. Application of deep learning for object detection. *Procedia Computer Science*, 132:1706 – 1717, 2018.

[2] S. Aslam. Omnicoreagency: Instagram statistics facts for 2019, 2019. Instagram by the numbers: usage stats, demographics and facts you need to know.

[3] L. Ballan, M. Bertini, A. Del Bimbo, A. M. Serain, G. Serra, and B. F. Zaccone. Combining generative and discriminative models for classifying social images from 101 object categories. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1731–1734, Nov 2012.

[4] E. Chatzilari, S. Nikolopoulos, I. Patras, and I. Kompatsiaris. Leveraging social media for scalable object detection. *Pattern Recognition*, 45(8):2962 – 2979, 2012.

[5] Y. G. Cinar, S. Zoghbi, and M. Moens. Inferring user interests on social media from text and images. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1342–1347, Nov 2015.

[6] M.A. Ranzato P. Perona F. Li, M. Andreetto. Caltech101 image dataset. 2003.

[7] Image classification: car, motorbike, bicycle, 2012. https://www.kaggle.com/c/image-classification2/data.

[8] Competition distinguish dogs from cats, 2014. https://www.kaggle.com/c/dogs-vs-cats/overview.

[9] D. Keyes. Instagram rolls out shoppable posts for more merchants, 2017. https://www.businessinsider.com/instagram-rolls-out-shoppable-posts-for-more-merchants-2017-10.

[10] A. Maksimov. Docker container with jupyter, matplotlib, pandas, tensorflow, keras and opencv, 2019. https://hub.docker.com/r/amaksimov/python_data_science/.

[11] A. Rosebrock. Multi-label classification with keras, 2018. https://www.pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/.

[12] C. Segalin, A. Perina, M. Cristani, and A. Vinciarelli. The pictures we like are our image: Continuous mapping of favorite pictures into self-assessed and attributed personality traits. *IEEE Transactions on Affective Computing*, 8(2):268–285, April 2017.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.