DOI:10.48011/asba.v2i1.1599

Visão Computacional e Aprendizado de Máquina para Classificação de Componentes em Alimentadores de Subestações Elétricas

Bruno Alberto Soares Oliveira *,**
Abílio Pereira de Faria Neto **
Roberto Márcio Arruda Fernandino **
Rogério Fernandes Carvalho **
Diego de Proença Costa ***
Frederico Gadelha Guimarães *

* *Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte-MG, (e-mail: {brunoalbertobambui, fredericoguimaraes} @ufmg.br)*
** *SVA Tech, Belo Horizonte-MG, (e-mail: {abilio.faria, roberto.fernandino, rogerio.fernandes} @svatech.com.br)*
*** *CPFL Energia, Campinas-SP, (e-mail: diegocosta@cpfl.com.br)*

Abstract: Nowadays, manual monitoring for the management of works in electric power substations has become recurrent on a daily basis, as there is a constant growth in society's demand for energy. That said, there are opportunities to automate such processes using artificial intelligence and, therefore, the objective of this work is to make a comparison between four architectures of convolutional neural networks: DenseNet, Inception, ResNet and SqueezeNet, aiming to solve a problem of classification of elements that make up feeders in power substations. 32 photos of different feeders were captured in the field and, later, data augmentation techniques were applied, aiming to increase the number of images. The models were trained using transfer learning and both obtained results close to 100% in test set, being possible to conclude that, for the proposed problem, there was no significant difference between their assertiveness.

Resumo: Hodiernamente o monitoramento manual para a gestão de obras em subestações de energia elétrica vem se tornando recorrente no dia a dia, visto que há um constante crescimento na demanda da sociedade pela energia. Dito isso, surgem oportunidades de automatização de tais processos utilizando a inteligência artificial e, sendo assim, o objetivo deste trabalho é realizar um comparativo entre quatro arquiteturas de redes neurais convolucionais: DenseNet, Inception, ResNet e SqueezeNet, para resolver um problema de classificação de elementos que compõem alimentadores em subestações de energia. Foram capturadas em campo 32 fotos de diferentes alimentadores e, posteriormente, foi aplicado técnicas de aumento de dados, para aumentar a quantidade de imagens. Os modelos foram treinados utilizando aprendizado por transferência e ambos obtiveram resultados próximos a 100% no conjunto de teste, sendo possível concluir que, para o problema proposto, não houve diferença significativa entre a assertividade destes.

Keywords: Management of works; Computational Intelligence; Monitoring; Image processing; Convolutional Neural Networks; Computer vision.

Palavras-chaves: Gestão de obras; Inteligência Computacional; Monitoramento; Processamento de Imagens; Redes Neurais Convolucionais; Visão Computacional.

1. INTRODUÇÃO

Atualmente, o monitoramento das distribuidoras de energia que gerenciam as atividades das obras nas subestações de distribuição tem sido realizado de forma presencial e manual, estando suscetível a erros técnicos ou até mesmo à má fé do funcionário. Regularmente, em curtos períodos definidos pela empresa, os responsáveis devem se deslocar a cada subestação para realizar o processo de monitoramento e avanço das obras.

O grande volume de empreendimentos realizados pelo grupo da CPFL Energia demanda cada vez mais recursos humanos para realizar a gestão de obras. Anualmente o time de engenharia realiza a gestão de mais de 85 obras, das quais aproximadamente 20% (por volta de 15 obras) são de instalações de alimentadores de 15kV, que apresentam um custo médio de R$ 320 mil por obra. Os alimentadores são conjuntos de elementos com funções específicas no sistema elétrico, que permitem a

⋆ A atual pesquisa é financiada pelo Programa P&D ANEEL (PD-00063-3073/2020) através da CPFL Energia e com parceria da empresa SVA Tech.

composição da subestação em módulos. Alimentadores de subestações têm como principais componentes: painel, transformadores de corrente, transformadores de potência, chaves seccionadoras e isoladores. A Figura 1 apresenta dois exemplos de alimentadores.

Figura 1. Exemplo de dois alimentadores.

Alguns trabalhos encontrados na literatura demonstram a aplicação da tecnologia em resolver problemas presentes em subestações de energia (Oliveira et al., 2019a) e também auxiliando no monitoramento da gestão de obras. Na pesquisa desenvolvida por Lin et al. (2016) foi discutido que o aumento de imagens de infravermelho no sistema elétrico de subestações apresenta um novo desafio, exemplificando a forma como a avaliação do status dos dispositivos é feita tradicionalmente. Para superar esse problema, o trabalho de Lin et al. (2016) propõe um método de análise automática de imagens de infravermelho. O algoritmo implementado primeiro segmenta imagens em superpixels e, em seguida, adota redes neurais convolucionais para classificação dos objetos. Foi alcançado bons resultados em comparação com outros métodos de treinamento testados.

No trabalho de Wang (2018) foi proposta uma nova abordagem de reconhecimento de estados baseado em redes neurais convolucionais de comutadores em subestações. Foi empregado aprendizado por transferência no modelo, em que os autores usaram o conjunto de imagens ILSVRC2012, retreinando o modelo com suas próprias imagens da subestação estudada. Em seus experimentos obtiveram resultados bastantes satisfatórios e concluíram que a abordagem proposta pode ser aplicada a subestação analisada, podendo reduzir o custo de operação caso a indústria decida implantar o sistema.

Para o gerenciamento adequado de obras e as revisões do plano durante a construção, é preciso que se entenda o status da obra em tempo real. Para auxiliar no desenvolvimento de soluções para a construção civil, Kim et al. (2018) propõem um método de aprendizado profundo objetivando reconhecer com precisão equipamentos de construção. Pela falta de imagens disponíveis dos objetos abordados, foi utilizado o aprendizado por transferência que classifica cinco classes: caminhão basculante, escavadeira, carregadeira, caminhão betoneira e rolo compactador. Os resultados atingiram 96,33% para a precisão média. O método proposto pode ser utilizado para inferir o contexto das operações de construção civil, gerando dados como o progresso da obra, produtividade e segurança.

De forma similar, Fang et al. (2018) desenvolveram um sistema para monitoramento de gestão de obras na construção civil, realizando a detecção automática de trabalhadores e escavadeiras em um determinado canteiro de obras. Foi utilizado o método R-CNN mais rápido que obteve ótimos resultados, detectando a presença de trabalhadores e escavadeiras com um alto nível de precisão (91% e 95%). De acordo com os autores, a precisão do método de aprendizado profundo proposto excede a dos métodos atuais na detecção de objetos em canteiros de obras.

No trabalho de Arabi et al. (2019) foi desenvolvida uma solução baseada em aprendizado profundo para detecção de equipamentos de construção, partindo de um loteamento vago até a obra totalmente concluída. A análise dos resultados confirma o bom desempenho da solução proposta com uma taxa de precisão acima de 90%. O presente estudo valida a praticidade de soluções de detecção de objetos baseadas em aprendizado profundo para cenários de construção. Além disso, a solução pode ser empregada para diversas finalidades, como monitoramento de segurança, avaliações de produtividade e decisões gerenciais.

Diante das pesquisas apresentadas e devido à padronização dos projetos, muitas atividades na gestão de obras tornam-se recorrentes e processuais, criando oportunidades para utilização de tecnologias de automação do processo de gestão, e, portanto, ganho de produtividade do recurso humano empregado. Logo, o objetivo desse estudo é realizar um comparativo entre arquiteturas de redes neurais convolucionais, implementando modelos de aprendizado de máquina que sejam capazes de classificar os principais elementos que compõem alimentadores em subestações de energia elétrica, para, futuramente, utilizar esse modelo treinado como parte de um sistema que servirá para otimizar as atividades de gestão de obras em subestações.

2. REFERENCIAL TEÓRICO

As aplicações no campo da inteligência artificial vêm crescendo consideravelmente nos últimos anos, potencializando a relação entre homens e máquinas (Oliveira et al., 2019b). Tanto no meio industrial quanto em pesquisas acadêmicas, profissionais de praticamente todas as áreas do conhecimento trabalham para a evolução dessa tecnologia (Oliveira et al., 2018). Uma das inúmeras subáreas da inteligência artificial é a visão computacional, que vem sendo ainda mais aprimorada.

A visão computacional pode ser entendida como um ramo da inteligência artificial que permite que máquinas consigam ver o mundo assim como os seres humanos, extraindo recursos a partir de imagens e auxiliando na tomada de decisão (Forsyth and Ponce, 2002). A visão computacional pode ser utilizada em várias tarefas, como no reconhecimento de imagem e vídeo, análise e classificação de imagens, recreação de mídia visual entre outras coisas. Os avanços em visão computacional e aprendizado de máquina foram evoluindo com o tempo, principalmente por meio das Redes Neurais Convolucionais (CNNs).

2.1 Redes Neurais Convolucionais

Uma rede neural convolucional, também chamada de ConvNet, é um algoritmo de aprendizado de máquina que recebe uma imagem como entrada e, a partir dela, consegue extrair recursos atribuindo importância (pesos e viéses aprendidos) a vários aspectos, sendo assim capaz de diferenciar objetos ali contidos (Krizhevsky et al., 2012). A estrutura de uma CNN foi inspirada de acordo com a organização do córtex visual dos seres humanos. Neurônios individuais respondem aos estímulos que correspondem a determinada região restrita do campo visual, conhecida como campo receptivo. O agrupamento desses campos se sobrepõem sendo capazes de cobrir toda a área visual (Kriegeskorte, 2015).

Diferentemente de técnicas convencionais de processamento digital de imagens que utilizam filtros previamente estabelecidos, as CNNs conseguem aprender de forma automática os melhores parâmetros dos extratores de recursos. Com isso, conseguem obter um ótimo desempenho em tarefas de classificação, detecção e segmentação de objetos, superando a precisão alcançada pelas técnicas convencionais de processamento digital de imagens.

A arquitetura de uma CNN é composta por camadas de convolução, camadas de pool e camadas totalmente conectadas. Cada camada de convolução possui uma quantidade de filtros de tamanhos determinados, que são responsáveis por extrair os recursos de alto nível da imagem de entrada. As camadas de pool possuem a função de reduzir o tamanho espacial do recurso envolvido, implicando em uma redução de dimensionalidade dos dados e consequentemente diminuir o recurso computacional empregado para processamento. Por fim, as camadas totalmente conectadas aprendem as combinações não lineares dos recursos, fazendo com que, através de uma função de ativação no final da rede, consiga classificar objetos.

Existe uma infinidade de possibilidades de disposição das camadas que compõe uma CNN. Quanto mais profunda for a arquitetura de uma CNN, maior será o gasto de processamento computacional empregado, ou seja, a arquitetura definida varia de acordo com o problema. Atualmente, encontra-se disponível algumas arquiteturas de CNNs que foram pré-treinadas em grandes conjuntos de imagens e obtiveram um desempenho bastante satisfatório. Algumas dessas arquiteturas de CNNs conhecidas são a DenseNet, Inception, ResNet e SqueezeNet.

DenseNet As redes convolucionais densamente conectadas surgiram do trabalho de Huang et al. (2017), em que foi proposta uma rede que, além de profunda, também é precisa e eficiente. Os autores acreditavam que caso as conexões entre as camadas fossem mais curtas, o treinamento da rede seria melhor ajustado. Com isso, a implementação da arquitetura foi definida com a premissa de que para cada camada, com os mapas de recursos obtidos pelas camadas anteriores, estes também seriam usados como entrada em todas as demais camadas subsequentes.

A arquitetura DenseNet possui algumas vantagens em relação as arquiteturas concorrentes: (1) a rede é capaz de minimizar o problema da fuga do gradiente, (2) fortalece a propagação dos recursos obtidos através dos operadores de convolução, (3) reutiliza tais recursos e (4) reduz substancialmente o número de parâmetros da rede. Conforme apresentado por Huang et al. (2017), a arquitetura DenseNet foi avaliada em quatro tarefas de *benchmark* de classificação de objetos (CIFAR10, CIFAR-100, SVHN e ImageNet), obtendo melhorias significativas em relação ao estado da arte em grande parte deles, exigindo menos memória e desempenho computacional.

Inception A arquitetura de CNN Inception foi apresentada pela primeira vez no trabalho de Szegedy et al. (2016). Os autores concordavam que quanto mais profunda a rede e quanto maior o poder de processamento computacional, maior seriam os ganhos de precisão na grande maioria das tarefas. Porém, discutiam que a eficiência computacional e o número limitado de parâmetros ainda são configurações que poderiam ser ajustadas e melhoradas, principalmente ao se trabalhar com um grande conjunto de dados e com técnicas de processamento digital de imagens.

Dito isso, na arquitetura proposta por Szegedy et al. (2016) foi explorado um meio de utilizar os recursos computacionais de forma eficiente, através de convoluções adequadamente fatoradas e regularização agressiva. A arquitetura de CNN Inception foi testada no desafio de classificação do ILSVRC 2012, obtendo um erro de 21,2% no top-1 da competição, superando o estado da arte da época.

ResNet As redes neurais residuais propostas no trabalho de He et al. (2016) apresentam uma arquitetura de CNN estruturada no aprendizado residual, facilitando o treinamento de redes profundas. Os autores reformularam explicitamente as camadas de aprendizado utilizando funções residuais na entrada das camadas, ao invés de empregar funções de ativação desconexas. Com isso, foi possível mostrar que a arquitetura de CNN ResNet é mais fácil de ser otimizada e é capaz de obter uma precisão elevada mesmo com uma profundidade consideravelmente maior.

A ResNet foi avaliada no conjunto de dados ImageNet, obtendo um erro de 3,57% no conjunto de teste, conquistando o primeiro lugar ILSVRC 2015 para a tarefa de classificação de objetos. Atualmente, é considerada uma das principais arquiteturas de CNNs, sendo empregada em problemas de classificação e detecção de objetos, tanto em imagens quanto em vídeos.

SqueezeNet A grande maioria das arquiteturas de CNNs propostas se concentram em melhorar principalmente a precisão. Dito isso, no trabalho de Iandola et al. (2016) foi proposta uma arquitetura de CNN bem menos profunda que as concorrentes, chamada SqueezeNet. Com precisão equivalente, a SqueezeNet oferece pelo menos três vantagens em relação as demais arquiteturas: (1) redes menores requerem menos comunicação entre servidores durante o treinamento utilizando processamento distribuído; (2) redes enxutas exigem menor largura de banda de modo a exportar o modelo treinado em aplicações que necessitam deste recurso; (3) redes menores são mais viáveis de serem empregadas em aplicações que operam em dispositivos embarcados e *hardwares* que possuem pouca memória.

A arquitetura de CNN SqueezeNet atinge uma precisão com um valor aproximado dos alcançados pelas arquiteturas concorrentes no conjunto de imagens ImageNet e,

paralelo a isso, ainda possui cerca de 50x menos parâmetros que a AlexNet, uma outra arquitetura de CNN. Além disso, com as técnicas utilizadas para a compactação do modelo, a SqueezeNet pode ser compactada de modo que ocupe apenas 0,5MB de espaço em disco, cerca de 510x a menos que a AlexNet.

3. METODOLOGIA

Cada passo da execução deste trabalho pode ser observado de uma maneira geral a seguir.

(1) Coleta das imagens em campo e preparação do banco.
(2) Divisão das imagens em treino, validação e teste, variando aleatoriamente as divisões.
(3) Aplicação de técnicas de aumento de dados.
(4) Implementação das arquiteturas de CNNs: DenseNet, Inception, ResNet e SqueezeNet.
(5) Realização de experimentos e análise dos resultados.

3.1 Coleta e preparação de dados

Para o desenvolvimento desta pesquisa foram capturadas 32 fotos de diversos alimentadores em diferentes subestações da empresa CPFL Energia. Com isso, foram criados cinco conjuntos de imagens diferentes sendo que, para cada um dos conjuntos, as 32 imagens foram divididas aleatoriamente entre o conjunto de treino, validação e teste. Tal divisão é apresentada na Tabela 1.

Tabela 1. Exemplo de aleatoriedade do conjunto de imagens para o primeiro experimento.

CONJUNTO DE IMAGENS 1	
Treino	
base	[1, 5, 8, 10, 11, 12, 13, 16, 17, 18, 21, 22, 23, 24, 25, 28, 30]
grade	[2, 3, 4, 5, 6, 7, 9, 12, 13, 17, 18, 19, 22, 23, 24, 25, 26, 27, 28, 31]
painel	[2, 4, 5, 6, 8, 9, 10, 11, 12, 15, 17, 19, 20, 22, 24, 25, 26, 28, 31, 32]
tc	[1, 2, 4, 6, 7, 8, 10, 11, 12, 14, 15, 16, 20, 21, 22, 28, 29, 30, 31, 32]
Validação	
base	[14, 19, 26]
grade	[1, 21, 29, 32]
painel	[1, 13, 14, 16, 18]
tc	[5, 13, 18, 19, 25]
Teste	
base	[15, 20, 27]
grade	[10, 14, 15, 16, 20]
painel	[3, 7, 21, 23, 27, 29, 30]
tc	[3, 9, 17, 23, 24, 26, 27]

A Tabela 1 exemplifica a aleatoriedade imposta na divisão dos dados de treino, validação e teste, para o primeiro conjunto de imagens gerado. A mesma aleatoriedade também foi aplicada nos outros quatro conjuntos criados. A separação das imagens originais, previamente realizada antes da aplicação das operações de aumento de dados, é justificada pelo fato de que não se deve permitir que os modelos fiquem super ajustados durante a etapa de treinamento, pois, caso isso ocorra, o modelo pode não conseguir generalizar bem para o conjunto de teste. A Figura 2 apresenta o *pipeline* de divisão entre os conjuntos de dados e o uso das operações de aumento de dados.

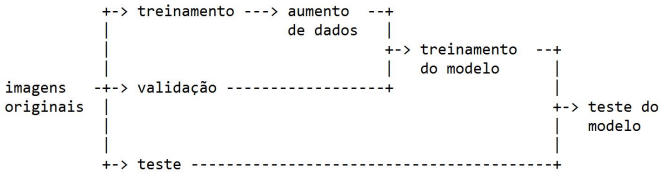

Figura 2. *Pipeline* do processo de divisão das imagens.

Manualmente, para cada uma das 32 fotos capturadas, foram geradas 20 imagens para cada uma das quatro classes de objetos, por meio de recortes de tamanhos aleatórios, mas sempre mantendo o objeto pertencente a determinada classe em questão na parte interna do corte. Além disso, para cada um dos 20 recortes, foi empregada a operação de inversão horizontal em cada cópia do corte, tendo ao final da aplicação do aumento de dados 40 imagens a partir da original.

É possível perceber na Tabela 1 que a classe base possui 23 imagens, e as classes grade, painel e tc, possuem 29, 32 e 32 imagens, respectivamente. Isso se deu pelo fato de que em algumas fotos capturadas em campo não foi possível gerar os cortes para todas as quatro classes na referente imagem. Para as classes painel e tc, em que ambos os objetos se encontram centralizados no alimentador, os mesmo estão presentes nas 32 fotos capturadas.

A Figura 3 ilustra exemplos de imagens com a aplicação do aumento de dados que foram utilizadas neste estudo.

Figura 3. Imagens contidas no conjunto de dados.

São quatro classes de objetos abordadas: base, painel, transformador de corrente (tc) e transformador de potência (tp). A classe “tp” foi chamada de “grade” neste trabalho pelo fato de que o objeto que pode ser visto na

imagem é uma grade, e não o "tp" em questão. Na prática, a grade instalada no alimentador serve para proteger o tp de pássaros que possam vir a danificar o sistema. A Figura 3 apresenta dois exemplos de objetos para cada uma das quatro classes: base, painel, tc e tp (grade), nessa respectiva ordem. Por fim, a Tabela 2 apresenta a quantidade de imagens por classe para cada um dos cinco conjuntos de treino, validação e teste.

Tabela 2. Quantidade de imagens contidas em cada um dos cinco conjuntos de dados.

	Treino	**Validação**	**Teste**
base	680	120	120
grade	800	160	200
painel	800	200	280
tp	800	200	280

3.2 Implementação das arquiteturas de CNNs

Realizada a etapa de preparação dos cinco conjuntos de dados a serem utilizados nos experimentos, foi efetuada a implementação das arquiteturas de CNNs escolhidas. O ambiente empregado para a codificação dos métodos foi o Google Colab, que é um serviço em nuvem para o desenvolvimento de algoritmos de aprendizado de máquina na linguagem de programação Python (Bisong, 2019). O Google Colab tem algumas limitações em sua versão gratuita, fazendo com que não fosse possível executar um maior número de experimentos e utilizar um conjunto de imagens mais robusto. Apesar disso, sua escolha é justificada pelo fato do mesmo oferecer suporte a GPU.

Para a implementação das arquiteturas das CNNs foi utilizada a ImageAI, que é uma biblioteca para a linguagem de programação Python que foi desenvolvida para capacitar desenvolvedores, pesquisadores e estudantes a criarem aplicativos e sistemas com recursos independentes de aprendizado profundo e visão computacional (Moses and Olafenwa, 2018). Suporta uma lista de algoritmos de aprendizado de máquina para classificação e detecção de objetos em imagens, dentre eles os utilizados neste trabalho: DenseNet, Inception, ResNet e SqueezeNet.

Com os materiais disponíveis as arquiteturas de CNNs foram implementadas respeitando sua topologia conforme descrita na literatura, quanto a disposição dos neurônios nas camadas de convolução e seus respectivos valores de hiperparâmetros, obedecendo a disposição dos elementos da rede, além do ajuste na etapa de extração de recursos.

3.3 Experimentos

Implementada as arquiteturas de CNNs e para um comparativo mais justo entre essas foi fixado o número de alguns hiperparâmetros durante o treinamento em cada modelo para cada experimento. Em todos os casos foi utilizado o aprendizado por transferência, um número de épocas de 100 vezes e o tamanho do lote como 32.

O aprendizado por transferência pode ser definido como a prática em utilizar o conhecimento adquirido ao resolver determinado problema, aplicando-o a um problema diferente. Nos experimentos realizados foi adotado um modelo pré-treinado no conjunto de dados ImageNet, diferente para cada arquitetura de CNN, uma vez que cada uma delas possui uma topologia própria. A vantagem da utilização desse tipo de aprendizado é que os pesos de cada neurônio não se inicializam aleatoriamente, diminuindo o tempo do treinamento. Nenhuma camada foi congelada, sendo todas retreinadas para o novo domínio do problema.

O número de épocas é um hiperparâmetro que define o número de vezes que o algoritmo de aprendizado funcionará durante todo o conjunto de dados de treinamento. Já o tamanho do lote (batch_size) é um hiperparâmetro que define o número de amostras (imagens) a serem analisadas antes de atualizar os parâmetros internos do modelo.

Ao fim de cada época do treinamento é avaliada no conjunto de validação a acurácia obtida e, em seguida, se salva o modelo da primeira época. Caso o valor da acurácia seja superior ao encontrado na época anterior, o modelo é novamente salvo contendo os novos parâmetros obtidos no treinamento daquela determinada época. Esse processo é repetido até que se complete o número de épocas previamente estabelecido.

Por fim, é feito o teste com o último modelo salvo em imagens jamais vistas durante o treinamento, que haviam sido previamente separadas na divisão aleatória entre os conjuntos de treino, validação e teste. A importância da divisão dos dados em três conjuntos diferentes está no fato de que, ao se empregar essa metodologia, seja possível avaliar com uma melhor precisão a capacidade que o modelo tem em classificar corretamente novas imagens do mundo real.

Em síntese, no primeiro experimento com o primeiro conjunto de dados que foi divido aleatoriamente, cada um dos quatro modelos foram treinados, validados e testados. Em seguida, o mesmo processo foi repetido utilizando os outros quatro conjuntos de imagens, em que cada um deles possui uma divisão diferente para os conjuntos de treino, validação e teste.

4. RESULTADOS E DISCUSSÃO

Os resultados apresentados se baseiam em duas premissas: (1) no modelo salvo na época em que o mesmo convergiu o obteve o maior valor de acurácia perante o conjunto de validação; (2) na quantidade de acertos para cada uma das classes nas imagens contidas no conjunto de teste. Para cada um dos cinco conjuntos de dados diferentes, tomou-se o cuidado para que o conjunto de teste tivesse 120 imagens da classe base, 200, 280 e 280 para as classes grade, painel e tc, respectivamente.

De uma maneira geral, percebeu-se que o tempo de treinamento da arquitetura de CNN SqueezeNet foi bem inferior, o que é justificado por possuir uma arquitetura mais enxuta, para aplicações que possuem uma menor capacidade de *hardware*, como dispositivos embarcados, por exemplo.

Por causa do aprendizado por transferência, uma vez que decidiu-se utilizar modelos pré-treinados no conjunto de dados ImageNet, o número de épocas em que os modelos (independente das arquiteturas) convergiram foi baixo, especialmente para o caso do DensetNet e Inception, que são arquiteturas de CNNs mais robustas. Caso os pesos da rede tivessem sido inicializados aleatoriamente, acredita-

se que o número de épocas para a conversão dos modelos teria sido bem maior.

Em número de acertos, independente da divisão aleatória dos conjuntos de treino, validação e teste, ambas as arquiteturas de CNNs se comportaram muito bem, com uma acurácia de quase 100%. Além disso, constatou-se que para os erros obtidos, a probabilidade da classe com maior valor foi, quase sempre, aproximadamente 40%, ou seja, o modelo obteve duas probabilidades (uma para a classe correta e outra para a classe a qual ele errou) bem próximas e baixas, o que ocasionou o erro da classificação.

4.1 Primeiro Experimento

A Tabela 3 apresenta os resultados obtidos no primeiro experimento para cada uma das arquiteturas de CNNs.

Tabela 3. Primeiro experimento.

	DenseNet	**Inception**	**ResNet**	**SqueezeNet**
Época	4	5	17	5
base	120	120	120	120
grade	200	200	200	200
painel	280	280	280	280
tc	280	280	280	280

É possível perceber na Tabela 3 o número baixo de épocas que as arquiteturas de CNNs DenseNet, Inception e SqueezeNet precisaram durante o treinamento para que elas conseguissem obter uma alta acurácia nas imagens do conjunto de validação. Na etapa de teste dos modelos, todos os quatro atingiram 100% de acerto para as imagens contidas no conjunto de teste, neste primeiro experimento.

4.2 Segundo Experimento

A Tabela 4 apresenta os resultados obtidos no segundo experimento para cada uma das arquiteturas de CNNs.

Tabela 4. Segundo experimento.

	DenseNet	**Inception**	**ResNet**	**SqueezeNet**
Época	8	4	13	34
base	120	120	120	120
grade	200	200	200	200
painel	280	280	280	267
tc	280	280	278	280

Nas informações ilustradas na Tabela 4 é possível observar os resultados que, para as arquiteturas de CNNs DenseNet, Inception e ResNet, os valores de acertos obtidos se assemelham bastante, chegando a um valor de 100% de classificações corretas para o DenseNet e o Inception. A arquitetura de CNN SqueezetNet teve alguma dificuldade em classificar objetos da classe painel, tendo acertado 267 em 280 imagens possíveis. Além disso, a SqueezetNet demorou 34 épocas para encontrar um ótimo valor de acurácia no conjunto de validação, número bem maior em relação aos obtidos pelas demais arquiteturas de CNNs.

Ao se analisar especificamente as probabilidades previstas das imagens que foram classificadas incorretamente pela arquitetura de CNN SqueezetNet neste segundo experimento, foi visto que não foram todas as imagens que obtiveram maior probabilidade prevista em apenas uma determinada classe, ou seja, o objeto "painel" não foi confundido apenas com o objeto "base", por exemplo. As imagens classificadas erroneamente tiveram em média como maior probabilidade um valor de aproximadamente 33% para o objeto classificado (podendo ser base, grade ou tc) e um valor aproximado para a classe painel.

4.3 Terceiro Experimento

A Tabela 5 apresenta os resultados obtidos no terceiro experimento para cada uma das arquiteturas de CNNs.

Tabela 5. Terceiro experimento.

	DenseNet	**Inception**	**ResNet**	**SqueezeNet**
Época	2	10	14	2
base	120	120	120	118
grade	200	200	200	200
painel	280	280	280	280
tc	280	280	280	280

O número de acertos das arquiteturas de CNNs propostas apresentados na Tabela 5 demonstram a viabilidade em se utilizar qualquer uma das arquiteturas apontadas para o problema de classificação proposto. No terceiro experimento realizado praticamente não houve diferença na assertividade dos modelos, exceto para o SqueezeNet, que classificou erroneamente como painel duas imagens que pertenciam a classe base.

4.4 Quarto Experimento

A Tabela 6 apresenta os resultados obtidos no quarto experimento para cada uma das arquiteturas de CNNs.

Tabela 6. Quarto experimento.

	DenseNet	**Inception**	**ResNet**	**SqueezeNet**
Época	4	5	18	6
base	120	120	120	120
grade	173	200	200	200
painel	280	264	254	280
tc	120	280	280	280

No quarto experimento se observou o baixo número de épocas gastas durante o treinamento para que os modelos convergissem, exceto a arquitetura de CNN ResNet, que precisou de um número maior de épocas para que a mesma conseguisse obter um alto valor de acurácia na etapa de validação. Houve uma diferença de assertividade em relação as arquiteturas de CNNs e as classes, exceto a arquitetura de CNN SqueezeNet, que obteve uma acurácia de 100% em todas as imagens do conjunto de teste. A arquitetura de CNN DenseNet não conseguiu classificar o objeto "grade" em 27 das 200 imagens possíveis. Já a arquitetura de CNN Inception classificou de maneira incorreta 16 imagens que pertenciam a classe "painel". Por fim, também de forma equivocada e com um número ainda maior, a arquitetura de CNN ResNet classificou erroneamente 26 imagens da classe "painel".

4.5 Quinto Experimento

A Tabela 7 apresenta os resultados obtidos no quinto experimento para cada uma das arquiteturas de CNNs.

Na Tabela 7 é possível verificar que as arquiteturas de CNNs abordadas obtiveram uma acurácia próxima a 100%

Tabela 7. Quinto experimento.

	DenseNet	Inception	ResNet	SqueezeNet
Época	7	4	12	11
base	120	118	120	120
grade	200	200	200	200
painel	280	280	278	280
tc	280	280	280	280

para o conjunto de teste no quinto experimento. A arquitetura de CNN Inception errou em apenas duas imagens que deveriam ter sido classificadas como "base", mas que foram rotuladas como "painel". Já a arquitetura de CNN ResNet classificou erroneamente também dois objetos mas, desta vez, pertencentes a classe "painel". Quanto ao número de épocas necessárias para que os modelos convergissem, observou-se que foi gasto um número baixo para todas as arquiteturas de CNNs empregadas.

5. CONCLUSÃO

Neste trabalho foi realizado um estudo comparativo entre quatro arquiteturas de CNNs: DenseNet, Inception, ResNet e SqueezeNet, aplicadas a um problema de classificação de quatro elementos que compõem alimentadores em subestações de energia elétrica. Foram capturadas 32 fotos em campo de diferentes alimentadores em subestações e, posteriormente, foi aplicada operações de aumento de dados para obter um conjunto maior de treinamento. Foi gerado aleatoriamente cinco diferentes conjuntos de imagens, cada um deles divididos em conjuntos de treino, validação e teste.

Independente da arquitetura de CNN empregada, conclui-se que os modelos obtidos durante os experimentos possuem uma alta assertividade para o problema em que foram aplicados, aproximando-se de 100%. Foi observado que existe um diferencial em relação a arquitetura de CNN SqueezeNet, que possui um menor tempo de treinamento e ocupa muito menos espaço em disco comparado com as demais, e ainda sim conseguiu obter uma assertividade semelhante nas imagens de teste.

Consequentemente, outras arquiteturas de CNNs podem vir a serem comparadas para o mesmo problema, além de, como trabalhos futuros, também pode-se aplicar outros problemas de classificação mais complexos e utilizá-los como objetos de comparação. O atual estudo será peça fundamental para a implementação futura de um método de detecção de objetos que também utiliza CNNs para a extração de recursos das imagens de entrada e, em seguida, encontra exatamente as posições de cada objeto na imagem.

REFERÊNCIAS

Arabi, S., Haghighat, A., and Sharma, A. (2019). A deep learning based solution for construction equipment detection: from development to deployment. *arXiv preprint arXiv:1904.09021*.

Bisong, E. (2019). *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer.

Fang, W., Ding, L., Zhong, B., Love, P.E., and Luo, H. (2018). Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach. *Advanced Engineering Informatics*, 37, 139–149.

Forsyth, D.A. and Ponce, J. (2002). *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K.Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Kim, H., Kim, H., Hong, Y.W., and Byun, H. (2018). Detecting construction equipment using a region-based fully convolutional network and transfer learning. *Journal of computing in Civil Engineering*, 32(2), 04017082.

Kriegeskorte, N. (2015). Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1, 417–446.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lin, Y., Sun, J., Li, C., Ma, Y., Geng, Y., and Chen, Y. (2016). Deep learning for intelligent substation device infrared fault image analysis. In *MATEC Web of Conferences*, volume 55, 03007. EDP Sciences.

Moses and Olafenwa, J. (2018). Imageai, an open source python library built to empower developers to build applications and systems with self-contained computer vision capabilities. URL `https://github.com/OlafenwaMoses/ImageAI`.

Oliveira, B.A.S., Assis, S., and Nolli, C. (2019a). Development of a prototype electrical energy monitoring system via internet/desenvolvimento de um protótipo de sistema de monitoramento de energia elétrica via internet. *Revista de Engenharia da Universidade Católica de Petrópolis*, 12(1), 48–61.

Oliveira, B.A.S., Assis, S., and Sasaki, R. (2019b). Aplicativo android para avaliar a qualidade da pulverização em condições de campo. *Revista Engenharia e Tecnologia Aplicada-UNG-Ser*, 2(1), 15–23.

Oliveira, B.A.S., Nametala, M.S., Castoril, B.R., Catalano, M.D., and Nametala, C.A.L. (2018). Avaliação de uma rede neural artificial como estimador temporal pluviométrico no sistema de abastecimento cantareira. *Revista de Informática Aplicada*, 14(1).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Wang, J. (2018). Deep learning based state recognition of substation switches. *AIP Conference Proceedings*, 1971(1), 040041.