Arquitetura de sistema em nuvem para apoio à implantação de Visão Computacional em linhas de produção na Indústria 4.0

Carlos Diego F. de Almeida [*] **Jean Phelipe de O. Lima** [*]
João Victor M. de Oliveira [*] **Raimundo Corrêa de Oliveira** [*]
Edson Farias Oliveira [**]

[*] *Escola Superior de Tecnologia, Universidade do Estado do Amazonas, AM (e-mail: {cdfa.eng16, jpdol.eng16, jvmdo.eng16, rcoliveira}@uea.edu.br).*
[**] *Digiboard Eletrônica da Amazônia, AM (eoliveira@digiboard.com.br)*

Abstract: Computer Vision Applications in Industry 4.0 fast becoming more and more frequent. A challenge in this scenario is to have a platform to support the implementation and deployment of these applications. This work proposes a cloud architecture to support the process of deploying computer vision systems in industry, supporting the phases of data ingestion, model training and model deployment of an Artificial Intelligence System. The system was implemented using the Function-as-a-Service cloud model through the AWS platform and evaluated according to Performance; Availability; and Cost. The results obtained were compared and validated with classical architectures, and the proposed architecture proved itself as an effective and efficient approach for the Industry.

Resumo: Aplicações de Visão computacional na Indústria 4.0 vêm se tornando cada vez mais frequentes. Um desafio neste cenário é dispor de uma plataforma de suporte à implementação e implantação dessas aplicações. Este trabalho propõe uma arquitetura em nuvem para apoiar o processo de implantação de sistemas de visão computacional na indústria, dando suporte às fases de *Data Ingestion*, *Model Training* e *Model Deployment* de um Sistema de Inteligência Artificial. O sistema foi implementado utilizando-se o modelo de computação em nuvem *Function-as-Service* através da plataforma AWS e avaliado de acordo com o Desempenho; Disponibilidade; e Custo. Os resultados obtidos foram comparados e validados com arquiteturas clássicas, e a arquitetura proposta demonstrou-se uma abordagem atrativa e eficiente para a Indústria.

Keywords: Computer Vision; Cloud Computing; Industry 4.0; Machine Learning.
Palavras-chaves: Visão Computacional; Computação em Nuvem; Indústria 4.0; Aprendizado de Máquina.

1. INTRODUÇÃO

A indústria 4.0, resultado da 4ª revolução Industrial, propõe um avanço no modelo de produção tradicional, Schwab (2017), adicionando tecnologias como Robótica, Inteligência Artificial (IA), Internet das Coisas (IoT), distribuição de informação, computação em nuvem, etc, de acordo com Bahrin et al. (2016). A IA, uma das principais tecnologias que compõem a Indústria 4.0, é responsável por proporcionar "inteligência" às máquinas, tornando-as hábeis a realizar trabalhos que antes apenas os seres humanos eram capazes de realizar. Uma aplicação recorrente de IA nas fábricas é a automação da inspeção visual, por meio da visão de máquina, Jain et al. (1995). Nesse processo são utilizados *inputs* e *outputs* digitais para manipular dispositivos mecânicos em um processo industrial, em que é necessário executar uma ação baseada na análise da imagem feita por um sistema de visão computacional. Esse método automático de inspeção visual é ideal, por exemplo, para verificação de componentes mal posicionados, erros de impressão de códigos de barras, entre diversas outras aplicações em processos industriais, como mostra Szeliski (2010). Assim, visão de máquina desempenha um papel crucial no controle de qualidade dos produtos e o faz com maior eficiência e eficácia que olhos humanos.

Pesquisas recentes, como as apresentadas em Voulodimos et al. (2018), indicam que as abordagens baseadas em *Deep Learning*, LeCun et al. (2015), têm obtido os melhores resultados para tarefas de visão computacional. *Deep Learning* é um ramo da inteligência artificial que pertence a subárea *Machine Learning* (ML) que, por sua vez, contempla um conjunto de técnicas capazes de fazer um computador adquirir conhecimento através de experiência, isto é, aprender a partir de dados históricos.

Existe então, pela natureza da técnica de *Deep Learning*, a necessidade de ser criada uma base de dados para ser utilizada durante o treinamento de um modelo inteligente que desempenhará a tarefa de visão computacional. Portanto, questões a respeito da criação dessa base de dados devem ser levadas em consideração, como número de imagens necessárias, iluminação adequada no ambiente de coleta,

necessidade de outros atributos preditores, etc. Todo esse processo de criação da base e treinamento do modelo inteligente deve acontecer fisicamente separado da linha de produção em que a visão computacional será empregada, evitando, dessa maneira, que a linha seja interrompida toda vez que se queira atualizar o modelo inteligente. Assim, surge a necessidade de um sistema que inclua: um ambiente para captura das imagens e outros dados, caso haja, do sistema de visão computacional; uma plataforma que utilize as imagens para treinar o modelo de IA; e, por fim, uma comunicação com a linha de produção para que o modelo inteligente seja atualizado quando treinado por novos dados.

Este trabalho propõe um sistema de apoio ao processo de implantação de visão computacional em linhas de produção. Para a demonstração da arquitetura proposta, foi utilizado o modelo de visão computacional descrito em Lima et al. (2019) para classificação de modelos de Placas de Circuito Impresso (PCBs). O sistema consiste em um dispositivo físico utilizado para captura de dados, onde são respeitados os requisitos necessários para criação de uma base que proporcione o melhor desempenho das técnicas de *Deep Learning*, e uma plataforma em nuvem, que recebe e armazena os dados amostrados, além de executar as rotinas de treino de modelos inteligentes que, quando criados ou atualizados, são destinados à implantação automática.

2. TRABALHOS RELACIONADOS

A utilização de visão computacional com técnicas de processamento digital de imagens e servidores locais para processamento e armazenamento de dados é uma prática comum há anos dentro da indústria. Com o advento da 4ª Revolução Industrial, as indústrias tendem a se adaptar a novas tecnologias. A utilização de *Deep Learning* para extração de características e reconhecimento de padrões em imagens e o uso de serviços em nuvem são consequências dessa adaptação. De fato, inteligência artificial e computação em nuvem são dois dos pilares da Indústria 4.0 e estão gradativamente substituindo as abordagens tradicionais.

A aplicação de *Deep Learning* em conjunto de dados proprietários na indústria de PCB não é novidade, como é mostrado em Júnior (2020), o autor construiu seu próprio conjunto de dados com 220 imagens de 18 placas diferentes em alta definição utilizando um microscópio USB. Em Singh and Joshi (2020), os autores também constroem a própria base de imagens de PCB e utilizam uma *webcam*, assim como neste trabalho. Em Lin (2021), o autor comprova a importância de um ambiente controlado e devidamente iluminado para a criação do *dataset*, bem como o tamanho máximo e posicionamento da placa e o campo de visão da câmera, fatores dos quais este trabalho leva em consideração. Em Ferreira and Jaimes (2020), os autores mostram que o uso de imagens capturadas em ambientes controlados resulta em inferências mais precisas comparadas àquelas em ambientes conturbados.

A utilização de *Cloud Computing* é vantajosa para a indústria em casos como o deste trabalho como é mostrado em Gupta et al. (2020), em que os autores propõem uma simples arquitetura para comunicação com serviços em nuvem na AWS e analisam suas vantagens e desvantagens. Em Hussain et al. (2020), os autores medem a eficiência dos *Web Services* da AWS para realização de inferências de seus modelos de *Machine Learning* armazenados na plataforma. E em Zeng et al. (2019), os autores vão além e propõem uma abordagem eficiente para realização de inferências em dispositivos embarcados na indústria.

3. CONSTRUÇÃO DE SISTEMAS DE *MACHINE LEARNING* ESCALÁVEIS

A Engenharia de Software consiste no estudo e aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de softwares, segundo IEEE (1993). Para o desenvolvimento de Sistemas de Inteligência Artificial não é diferente, contudo há algumas particularidades que apenas sistemas de *Machine Learning* apresentam. Dessa forma, surge a necessidade de um método de desenvolvimento de software que atenda as fases necessárias para uso comercial de Sistemas de IA (SIA).

Em muitos casos, profissionais de IA trabalham com ferramentas para desenvolvimento científico, como o Jupyter Notebook, por exemplo, que é o ambiente de desenvolvimento mais popular desses profissionais, como aponta Nguyen et al. (2019). Entretanto, esse tipo de desenvolvimento é útil na fase de validação do desempenho de modelos de *machine learning*, e até mesmo para criação de protótipos, mas uma metodologia completa projetada para tratar desde as fases iniciais do desenvolvimento de sistemas de *machine learning* até a fase de implantação e manutenção é necessária para que a aplicação se torne escalável.

A Figura 1 ilustra um *pipeline* de referência às atividades de desenvolvimento de um sistema de IA baseado em *machine learning*. Um sistema desenvolvido a partir desse fluxo de referência deve resultar em um SIA escalável. As etapas apontadas na Figura 1 são: *Problem Definition*, fase inicial, análoga a fase de levantamento de requisitos em desenvolvimento de softwares tradicionais; *Data Ingestion*, que inclui mineração, aquisição e levantamento de dados que serão utilizados na fase de treino/teste/validação do modelo de ML; *Data Preparation*, fase em que dados são manipulados, pré-processados e formatados como dados de entrada para os modelos de ML; *Data Segregation*, em que partições são efetuadas na base de dados seguindo alguma abordagem, como *holdout* ou *k-fold* por exemplo, resultando em subconjuntos de treino, teste e validação; *Model Train*, onde os exemplos do subconjunto de treino são apresentados ao modelo, para que sejam executados algoritmos de aprendizado; *Candidate Model Evaluation*, em que os resultados do treinamento são verificados a partir de métricas, como acurácia e f1-score, obtidas para os conjuntos de teste e validação (esta fase interage com a anterior até que se encontre um modelo com resultados satisfatórios); *Model Deployment*, em que o modelo de ML para realização da tarefa é escolhido e implantado em seu ambiente de aplicação; e, por fim, *Performance Monitoring*, em que o desempenho do modelo é continuamente monitorado. Uma iteração pode acontecer da etapa 8 para a 2 caso haja a necessidade de aumentar os exemplos da base de dados em busca de resultados melhores, ou ainda para a modificação do modelo, como inclusão ou exclusão de uma classe, por exemplo.

O sistema apresentado neste trabalho, dá suporte, sobretudo, às etapas *Data Ingestion*, *Model Training* e *Model Deployment*, além de permitir que aconteça a iteração do passo 8 para o passo 2. Os passos 1, 3, 4, 5, 6 e 8 são abordados em detalhes, para a mesma tarefa abordada neste trabalho, em Lima et al. (2019), o que o torna um trabalho complementar a este.

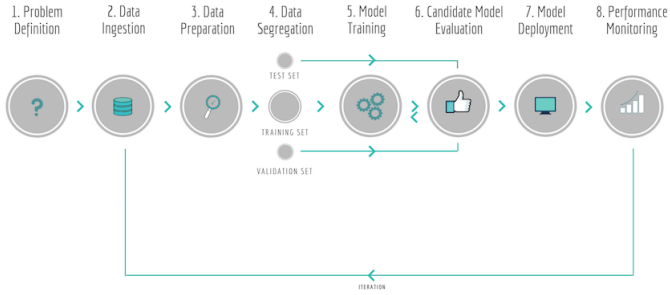

Figura 1. *Pipeline* de desenvolvimento de Sistema de *Machine Learning*

4. MATERIAIS E MÉTODOS

4.1 Arquitetura Geral do Sistema

Para que a utilização de um sistema de visão computacional baseado em *deep learning* torne-se viável em um ambiente fabril, é necessária uma infraestrutura que apoie os processos de criação, atualização, e uso do modelo de IA.

Considerando a natureza dinâmica das fábricas, é necessário que o sistema forneça suporte à inserção de novas informações na base. A atualização da base de dados deve ser realizada de maneira rápida, de forma que não ofereça atraso no processo produtivo da fábrica, porém, não deve apresentar perda de qualidade nos dados obtidos, visando o melhor desempenho do modelo gerado pelo treino da IA. O emprego de *machine learning* requer que o sistema possua a capacidade de executar treinos para geração de modelos inteligentes, atividade esta bastante custosa computacionalmente, portanto, faz-se necessário a disponibilidade de recursos computacionais robustos para acelerar tal processo. Levando todos esses fatores em consideração, foi criada uma arquitetura envolvendo hardware e software que viabiliza a realização e manutenção de todos esses processos de maneira eficiente. A arquitetura geral é apontada na Figura 2.

O Dispositivo de cadastro foi criado para atender a necessidade de um ambiente com condições necessárias para a captura de imagens de PCBs. O dispositivo foi planejado de forma que a imagem da placa pudesse ser obtida de maneira em que todas as características importantes para a geração do modelo inteligente fossem preservadas. Para alcançar esse objetivo, o equipamento conta com uma câmera com especificações ideais para a tarefa e em sua construção foi utilizado um método de iluminação que não gera reflexões, que será descrito na Subseção 4.2.

Foram utilizados serviços em nuvem, como indicado na Figura 2, pois o custo para implementar e realizar a manutenção de servidores locais é bastante alto. Ficam

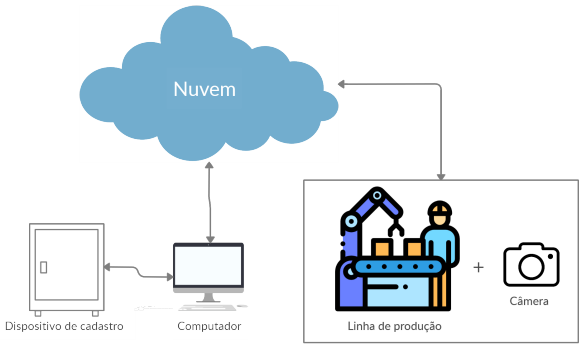

Figura 2. Arquitetura geral do sistema de classificação de PCBs.

hospedados na nuvem *webservices* que fornecem apoio a todos os processos que necessitam de armazenamento de dados e de poder computacional. O sistema oferece uma interface simples, porém eficaz, para a realização do cadastro de placas PCB, bastando apenas que o usuário acesse a plataforma através de um computador com acesso à rede e a câmera do Dispositivo de cadastro. O usuário fica a cargo de algumas simples responsabilidades, sendo elas: posicionar a placa de forma correta no dispositivo, efetuar a captura da imagem utilizando a plataforma, avaliar a imagem capturada e determinar se a foto possui qualidade satisfatória para que o cadastro seja concluído. A imagem e seus metadados são enviados para a nuvem assim que o usuário termina o cadastro.

O usuário pode iniciar o processo de treino para geração do novo modelo de IA contendo as novas informações assim que as imagens terminam de ser armazenadas. O modelo inteligente é enviado para a linha de produção logo após o término de sua geração onde imediatamente começa a ser utilizado para o reconhecimento de placas. Visando a maximização da acurácia do modelo, a linha de produção possui uma câmera com as mesmas especificações da utilizada no dispositivo de cadastro, para que a imagem capturada na linha possua a mesma qualidade das fotos utilizadas para treino.

4.2 Dispositivo de Cadastro

Percebe-se, através do *pipeline* de desenvolvimento de soluções baseadas em *Machine Learning* ilustrado pela Figura 1, que o passo 2 desse processo, *Data Ingestion*, tem importância fundamental para escalabilidade do sistema. Essa etapa é executada sempre que há atualização no modelo, seja pelo aumento de exemplos na base de dados para a tarefa contínua de melhorar os resultados alcançados pelo modelo quanto às métricas de avaliação utilizadas, ou pela inclusão de uma nova classe, por exemplo. Como o melhor desempenho de um modelo baseado em *Deep Learning* para visão computacional depende da utilização de imagens nítidas, sem ruídos, com boa iluminação, distância ideal para a câmera e com a menor interferência externa possível, existe a necessidade de criação de um ambiente controlado para que a fase de *Data Ingestion* seja conduzida da melhor maneira possível de tal forma a contribuir positivamente para a performance das etapas seguintes.

O ambiente de *Data Ingestion* desenvolvido neste trabalho foi denominado Dispositivo de Cadastro, cuja função é cadastrar os modelos de placa para atender a demanda da Indústria. Dessa forma, por meio do Dispositivo, o responsável pela produção das PCBs pode cadastrar modelos de placa conforme necessidade.

O Dispositivo de Cadastro foi projetado e construído a partir das necessidades de garantir uma base de dados consistente. Os parâmetros utilizados para o projeto foram: iluminação; distância para a câmera, abertura focal da câmera; tamanho máximo possível para um modelo de PCB. A iluminação deve ser controlada para favorecer a captura da imagem de modo que não haja reflexos de nenhuma espécie, por isso optou-se por vedar o dispositivo para que a iluminação externa não cause interferências, e, internamente, foram instaladas luzes brancas, para manter as cores reais das PCBs, em uma região lateral do dispositivo com os feixes luminosos direcionados às paredes do dispositivo, para que não haja focos de luz incidindo diretamente nas placas, tampouco feixes de luz emitidos diretamente à câmera, que fica posicionada no topo do dispositivo.

A distância para a câmera, abertura focal e tamanho máximo da PCB são parâmetros interdependentes e ditam as dimensões do dispositivo. Assim como apresentado em Lima et al. (2019), fez-se a utilização de uma *webcam* comum com lente de 50 mm, o que equivale a uma abertura focal de 40º, para a condução dos experimentos. A maior dimensão linear apresentada pelos modelos de PCB, também utilizados em Lima et al. (2019), é de 50 cm. A Figura 3 aponta esses parâmetros, em que x representa o tamanho máximo de PCB; h a distância entre a placa e câmera; e θ a abertura focal da câmera.

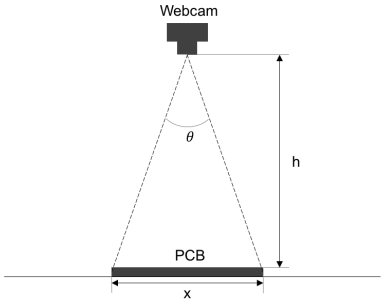

Figura 3. Representação dos parâmetros: Abertura Focal (θ); Distância para a câmera (h); e tamanho máximo de PCB (x).

Dessa forma, tem-se $\theta = 40^o$ e $x = 50cm$. Os segmentos de reta que delimitam a abertura focal formam um triângulo isósceles com x. A distância h parte θ ao meio, criando dois triângulos retângulos idênticos em que $x/2$ é cateto oposto a $\theta/2$. Assim, pela razão trigonométrica da tangente $tg(\theta/2) = \frac{x/2}{h}$, tem-se $h = \frac{x/2}{tg(\theta/2)}$ e, portanto, $h = \frac{50/2}{tg(40^o/2)} = \frac{25}{tg(20^o)} \approx 69cm$

A Tabela 1 relaciona os parâmetros e seus respectivos valores utilizados para construção do Dispositivo de Cadastro.

A Figura 4 ilustra o projeto mecânico do Dispositivo de Cadastro, em que é possível observar as características

Tabela 1. Valores dos parâmetros do projeto de construção do Dispositivo de Cadastro.

Parâmetro	Valor
Distância para a câmera	69 cm
Abertura Focal	40º
Tamanho máximo de PCB	50 cm

estabelecidas para o mesmo. A câmera está centralizada no topo, enquanto na parte de baixo há uma espécie de bandeja retrátil para que o operador insira a placa ergonomicamente. Um fato é que a existência de uma equivalência entre o ambiente de *Data Ingestion* e do ambiente de *deploy*, que neste caso corresponde a uma linha de produção, potencializa a performance do modelo de IA. Assim, é interessante que os parâmetros estudados para o projeto do Dispositivo de Cadastro sejam preservados durante a fase de implantação física do sistema.

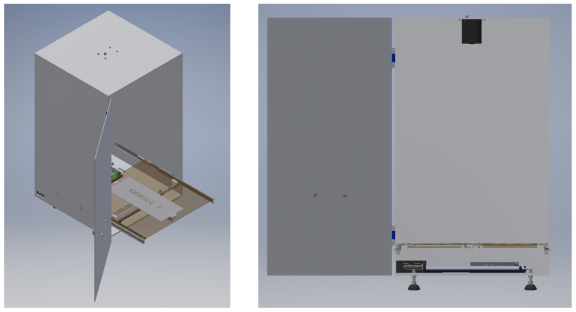

Figura 4. Projeto do Dispositivo de Cadastro de acordo com os parâmetros estabelecidos(Perspectiva Isométrica à esquerda e vista frontal à direita).

Por fim, o dispositivo precisa realizar uma comunicação com a plataforma *Cloud*, para que as imagens capturadas sejam transferidas. Para isso, criou-se uma interface web de comunicação entre Dispositivo de Cadastro, Plataforma em Nuvem e operador do sistema. Dessa maneira, um computador é acoplado ao Dispositivo de Cadastro; conectado a sua câmera interna; e, pela interface *web* implementada, o operador pode criar, digitalmente, um novo modelo de placa, realizar as capturas de imagens e, finalmente, enviar para a nuvem através de uma requisição, processo esse que será abordado em detalhes na subseção 4.3. O número de imagens capturadas para cada modelo de placa foi fixado em 10, de acordo com os experimentos apresentados em Lima et al. (2019), que indicam a necessidade de ao menos 10 imagens de placas posicionadas corretamente por modelo de placa para garantir o desempenho esperado do modelo de IA. A Figura 5 ilustra a Interface *web* de apoio a fase de *Data Ingestion*.

Na Figura 5, pode-se identificar o fluxo das telas para o processo de cadastro de uma nova placa. O primeiro passo (Figura 5.a) refere-se ao processo de criação lógica de um novo modelo de placa, em que o operador informa um código único do modelo de placa. A Figura 5.b aponta a interface de captura de imagens, em que o operador precisa coletar 10 imagens de placas devidamente posicionadas no interior do Dispositivo de Cadastro. Ainda na Figura 5.b, nota-se um exemplo de imagem de uma placa capturada já no dispositivo, construído de acordo com as especificações mencionadas. Por fim, a Figura 5.c ilustra a tela de confirmação do cadastro, em que as imagens são

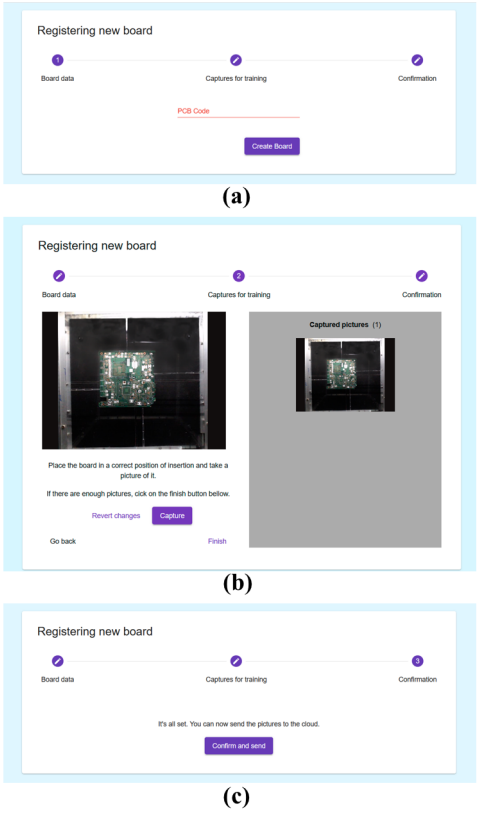

Figura 5. Interface *web* desenvolvida para comunicação entre Dispositivo de Cadastro, Plataforma em Nuvem e operador. **(a)** Criação de um novo modelo de PCB; **(b)** Captura de imagens; **(c)** Envio à nuvem.

submetidas à plataforma *cloud* após a confirmação por parte do operador.

4.3 Arquitetura de Nuvem

A implementação do sistema foi feita utilizando o modelo de computação em nuvem *Function-as-a-Service*, onde o código de apresentação e interação com o usuário roda no lado do cliente enquanto faz chamadas para APIs em que as regras de negócio estão encapsuladas. A plataforma escolhida para hospedar toda a aplicação foi a Amazon Web Services (AWS) por ser uma empresa já muito consolidada no mercado de computação em nuvem. A mesma arquitetura pode ser implementada em outras plataformas, como a IBM Cloud, Google Cloud e Microsoft Azure, pois possuem serviços relativamente equivalentes aos utilizados da Amazon Web Services. A arquitetura em nuvem é ilustrada na Figura 6.

A API foi desenvolvida utilizando recursos simples, porém extremamente poderosos da AWS, sendo eles: Funções Lambda, API Gateway, S3, Cognito, DynamoDB e IoT Core.

Cognito é uma ferramenta que fornece autenticação de usuários e dispositivos, permitindo a criação de um controle robusto de acesso a APIs. S3 é um serviço que disponibiliza espaço para manter arquivos, utilizou-se para armazenar os modelos inteligentes e as imagens usadas para o treino da IA. API Gateway gerencia requisições e respostas de APIs, todas as requisições HTTP passam por esse serviço e ele invoca outros para respondê-las. AWS Lambda são funções *stateless* que permitem que códigos sejam executados sem a necessidade de provisionar servidores, e é onde toda a lógica da aplicação reside. DynamoDB é o banco de dados NoSQL da AWS que foi utilizado para armazenar informações dos modelos de PCBs. IoT Core, que permite a conexão de dispositivos IoT com a nuvem da AWS através do protocolo MQTT, foi utilizado para a atualização automática do modelo inteligente em produção.

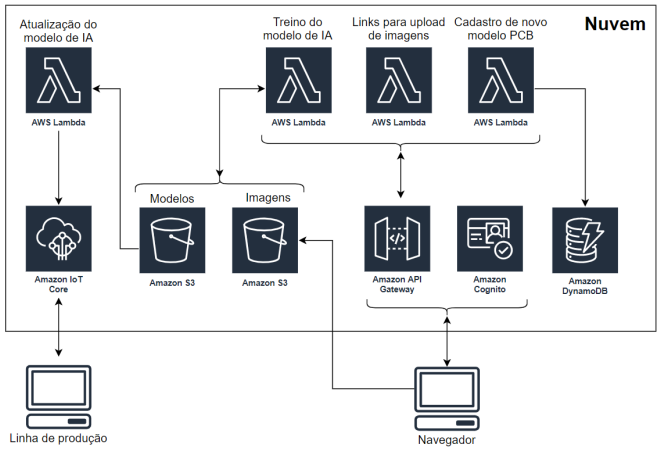

Figura 6. Arquitetura em nuvem para apoio a geração de modelos inteligentes para classificação de PCBs.

A API estará acessível na internet, portanto é necessário uma forma de controlar quem a acessa. A autenticação para uso da API será feita com o Cognito, utilizando o protocolo *Token-Based Authentication*. O usuário informa seus dados de *login* e o cognito retorna um *token* válido que será utilizado para realizar requisições a API.

Para dar início ao processo de cadastro, o usuário deve primeiramente criar o novo modelo de PCB no sistema inserindo seus dados de identificação. Logo após, a captura das 10 imagens necessárias para a geração do novo modelo inteligente pode ser iniciada. O usuário irá posicionar a placa no dispositivo de cadastro e irá controlar a captura das imagens utilizando a interface web, cabendo a ele constatar a boa qualidade de cada uma. Após a captura das imagens a interface web irá realizar o *upload* delas diretamente ao Bucket de imagens, utilizando links específicos que serão providenciados pela própria API.

Com as imagens devidamente validadas e carregadas na nuvem, o usuário pode iniciar o processo de treino da IA, através de uma requisição HTTP realizada pela interface web. A função lambda responsável por responder a essa requisição irá carregar todas as imagens contidas no Bucket de imagens e as utilizará no processo de geração do novo modelo inteligente, o qual será armazenado no devido Bucket.

Toda a comunicação feita entre a nuvem e a linha de produção dar-se-á por meio do AWS IoT Core, utilizando o protocolo MQTT. A segurança de acesso é feita a partir de certificados providenciados pelo próprio serviço. Quando um novo modelo inteligente é adicionado ao Bucket, uma função lambda é automaticamente acionada a partir de um

trigger. Essa função é responsável por publicar o link para download do novo modelo em um tópico MQTT específico o qual a linha de produção estará inscrita. A linha de produção inicia o download do arquivo imediatamente assim que recebe o link, ao fim desse processo, o modelo utilizado para o reconhecimento de placas é atualizado.

4.4 Sistema de treino da IA

Para a realização da tarefa de visão computacional proposta é necessário que um modelo inteligente seja gerado. Neste caso, utilizou-se como padrão a arquitetura de Redes Neurais Convolucionais (CNN) proposta em Lima et al. (2019), apontada na Figura 7 para classificação de placas de circuito impresso. Uma vez criada essa rede neural, a mesma é instanciada com pesos sinápticos aleatórios, em seguida, inicia-se o processo de treino supervisionado, em que exemplos são apresentados à rede e, após o processamento do exemplo, a saída obtida é comparada à saída desejada. A partir dessa comparação os pesos são ajustados com base no algoritmo *backpropagatioin*. A cada iteração, em que os todos os exemplos são apresentados à rede, tem-se uma época.

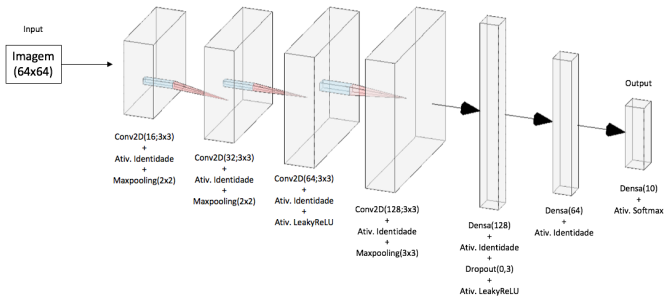

Figura 7. Arquitetura da CNN para a tarefa de classificação de PCBs.
Fonte: Lima et al. (2019)

A rotina de treino, hospedada no sistema *cloud*, é realizada sempre que há modificações na base de dados, seja por acréscimo ou decréscimo de modelos de PCB nela contidos. O número de épocas foi fixado em 20, que foi suficiente para convergência da curva de acurácia e de *loss* para todos os modelos de placa apresentados em Lima et al. (2019). Ao fim do treino, o novo modelo inteligente é gerado.

4.5 Avaliação do Sistema

Existem diversas formas de se avaliar um sistema baseado em *Cloud*. No contexto da aplicação apresentada, foram definidas como métricas de avaliação: desempenho, disponibilidade e custo. A seguir será apresentado como cada uma das análises avaliativas serão conduzidas:

Desempenho: O AWS Lambda é o serviço que o sistema descrito neste artigo utiliza para realizar o treino da IA. As funções lambdas permitem a execução de códigos sem a necessidade de provisionar e administrar servidores. O código ainda é executado em servidores, porém, todo o gerenciamento do servidor é feito pela AWS, sem a necessidade de interferência por parte do cliente. Para oferecer a escalabilidade adequada, a Amazon executa diversas cópias da mesma função em paralelo. A capacidade de CPU e outros recursos são proporcionais a quantidade de memória selecionada para a função, no sistema descrito neste artigo foi utilizado 10240 MB. Na subseção 5.1 é mostrado os resultados de desempenhos obtidos através de uma série de testes.

Disponibilidade: Disponibilidade é a probabilidade de um sistema estar disponível quando for necessária sua utilização. A *Amazon* fornece os dados de disponibilidade de seus serviços em seus devidos *Service Level Agreements* (SLAs).

A disponibilidade do sistema descrito neste artigo foi avaliada levando em consideração o que é informado no SLA de cada serviço utilizado. Os valores de disponibilidade podem ser encontrados na subseção 5.2.

Custo: O sucesso da infraestrutura em *Cloud* se deve eficiência do custo que os *Cloud providers* oferecem, principalmente em aplicações que possuam uso de computação de forma esporádica, pois o valor é calculado somente pelo tempo de computação utilizado, não é necessário que um servidor fique ligado se deteriorando enquanto aguarda ser usado. A não necessidade de investir em uma infraestrutura de hardware e software local estimula o surgimento de novos negócios, uma vez que diminui substancialmente a barreira de entrada.

Os *Cloud providers* abstraem para seus clientes todas essas preocupações, como são detentores de todos os recursos eles também assumem toda a responsabilidade de gerenciamento e manutenção dos mesmos, dessa forma, não se faz necessária a contratação de uma grande equipe especializada para a realização de tal tarefa, e também elimina a necessidade de se preocupar com espaço físico para acomodar toda a infraestrutura e gastos com energia elétrica.

A *AWS* encapsula todo o custo de operação, manutenção e gerenciamento dos seus serviços, portanto, a avaliação do custo foi feita em relação aos valores apresentados em seu site e podem ser conferidos na subseção 5.3.

5. RESULTADOS E DISCUSSÕES

5.1 Desempenho do sistema de Treino da IA em nuvem

Para análise de desempenho do sistema, foram utilizados o programa Apache JMeter, desenvolvido especialmente para medir a performance de aplicações web em testes de carga, e o serviço de *logs* da AWS, Cloud Watch. O aplicativo JMeter foi configurado para fazer 30 requisições sequencias intervaladas por 1 segundo para quantidades distintas de classes na base de dados, os resultados foram capturados utilizando o serviço Cloud Watch. Inicialmente, treinou-se com duas classes e em seguida esse número foi incrementado em 1 até atingir o total de 7 classes, assim, um total de 180 treinos fora realizados. Os resultados obtidos foram medidos quanto a duração do treino. Na Figura 8 são apontados os resultados médios obtidos em relação a quantidade de classes usadas para treino.

Através do gráfico exibido na Figura 8 é possível notar que mesmo com um número significativo de classes, o

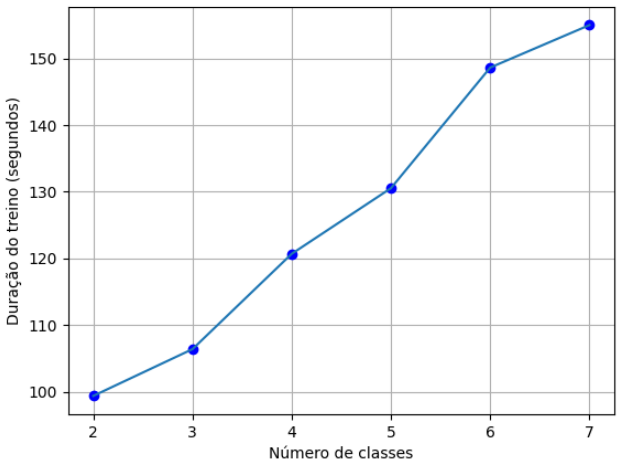

Figura 8. Desempenho médio do *AWS Lambda* em treino de IA para classificação de PCBs.

tempo necessário para geração do modelo inteligente ainda é pequeno.

5.2 Disponibilidade

A porcentagem de disponibilidade de cada serviço utilizado para o sistema proposto pode ser vista na Tabela 2.

Tabela 2. Disponibilidade dos serviços da AWS utilizados no sistema em nuvem para apoio a tarefa de classificação de PCBs. AWS (2021a)

Serviço	Disponibilidade mensal
API Gateway	99.95%
Cognito	99.90%
DynamoDB	99.99%
IoT Core	99.90%
Lambda	99.95%
Simple Storage Service (S3)	99,90%

Garantir o nível de disponibilidade apontado na Tabela 2 em uma infraestrutura *in-house* é uma tarefa mais complicada, alguns fatores devem ser levados em consideração. É necessário provisionar suporte 24/7, pois em caso de falha é essencial ter uma equipe pronta pra resolver o problema, além disso é recomendado possuir um acordo com o departamento responsável pela infraestrutura, onde é delimitado que serviços serão provisionados e em que condições, similar a um SLA.

5.3 Custo

As funcionalidades promovidas pelo sistema apresentado nesse artigo só se fazem necessárias na ocasião que um novo modelo de PCB precisa ser classificado, portanto, o custo total de operação da infraestrutura é praticamente irrisório. A seguir serão apresentados os preços de cada serviço utilizado.

O S3 foi utilizado para armazenar as imagens das PCBs necessárias para o treino da IA e os modelos inteligentes gerados. Na Tabela 3 é possível verificar que o custo de armazenamento desse serviço é mais interessante que montar um *datacenter*, e na Tabela 2 é possível observar sua disponibilidade.

Tabela 3. Definição de preço do S3 na região Norte da Virgínia. AWS (2021b)

Armazenamento	Custo
Primeiros 50 TB/mês	0,023 USD por GB
Próximos 450 TB/mês	0,022 USD por GB
Mais de 500 TB/mês	0,021 USD por GB

O AWS Lambda permite a execução de códigos de forma *serverless*. O custo desse serviço é calculado a partir de alguns fatores, sendo eles: número de invocações da função; tempo que o código levou para terminar de ser executado; e quantidade de memória selecionada para a função. A Tabela 4 contém exemplos do preço por 1 ms associados a diferentes tamanhos de memória.

Tabela 4. Definição de preço do *AWS Lambda* na região Norte da Virgínia. AWS (2021b)

Memória	Custo por 1 ms (USD)
128	0,0000000021 USD
2.048	0,0000000333 USD
6.144	0,0000001000 USD
8.192	0,0000001333 USD
10.240	0,0000001667 USD

O Amazon API Gateway, que permite a implementação rápida de APIs, recebe requisições HTTP e invoca outros serviços para respondê-las. O cliente paga apenas pelas chamadas de API recebidas e pela quantidade de dados transferidos para fora. Na Tabela 5 é possível ver a definição de preços do Amazon API Gateway.

Tabela 5. Definição de preço do *Amazon API Gateway* (REST) na região Norte da Virgínia. AWS (2021b)

Número de solicitações (por mês)	Preço (por milhão)
Primeiros 333 milhões	3,50 USD
Próximos 667 milhões	2,80 USD
Próximos 19 bilhões	2,38 USD
Mais de 20 bilhões	1,51 USD

Uma infraestrutura que forneça a disponibilidade e o desempenho que as empresas provedoras de serviços em nuvem prometem é bastante custosa, requer gastos com: hardware, licenças de softwares, espaço físico, um ambiente que promova um aumento da vida útil dos equipamentos e, principalmente, profissionais adequados para manter toda a infraestrutura. Na Tabela 6 é apontado o custo de manter uma equipe enxuta para planejar, implementar e gerenciar uma infraestrutura *in-house* que consiga fornecer um desempenho e disponibilidade minimamente aceitáveis.

Tabela 6. Salário de profissionais de TI nos EUA. indeed (2021)

Funcionário	Quantidade	Salário (por ano)
Administrador de banco de dados	1	$94,506
Administrador de sistema	1	$78,246
Administrador de rede	1	$69,698
Engenheiro de sistemas	1	$98,131
Engenheiro de redes	1	$96,736
Especialista em segurança da informação	1	$66,232
Gestor da tecnologia da informação	1	$93,529

Além dos gastos em recursos humanos, também é necessário manter a infraestrutura física do sistema, o que envolve custos com *racks* para organizar os dispositivos de armazenamento de dados, gastos com energia elétrica para manter o sistema ativo sempre, com refrigeração adequada, entre outros.

6. CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado um sistema de apoio a implantação de aplicações que façam uso de visão computacional em linhas de produção. O sistema fornece uma maneira eficiente e de baixo custo para realizar os retreinamentos dos modelos de IA, assim como colocá-los em produção com o menor tempo de inatividade da linha de produção, simplificando a tarefa de cadastro de novos objetos que precisam ser reconhecidos por essas aplicações.

O sistema consiste em um dispositivo especialmente planejado para captura de imagens com qualidade e o mínima de ruídos, e um *back-end* utilizando o modelo de computação em nuvem *Function-as-a-Service*, substituindo modelo antigo *in-house server infrastructure* que é mais custoso e demorado para ser implementado.

Atualmente o sistema necessita da intervenção humana para verificar se as imagens capturadas pelo dispositivo de cadastro estão aceitáveis ou não, além disso, a inferência é feita em um dispositivo embarcado posicionado na linha de produção. Dessa forma, é sugerido para trabalhos futuros a remoção ou diminuição da necessidade de intervenção humana no cadastro dos objetos e a realização das inferências diretamente na nuvem, diminuindo o custo em hardwares físicos locais para essa função.

REFERÊNCIAS

AWS (2021a). Disponibilidade aws. URL `https://aws.amazon.com/pt/legal/service-level-agreements/`.

AWS (2021b). Preços aws. URL `https://aws.amazon.com/pt/pricing/`.

Bahrin, M.A.K., Othman, M.F., Azli, N.H.N., and Talib, M.F. (2016). Industry 4.0: A review on industrial automation and robotic. *Jurnal Teknologi*. DOI: https://doi.org/10.11113/jt.v78.9285.

Ferreira, L.S. and Jaimes, B.R.A. (2020). Aplicação de visão computacional para automatização do processo de reconhecimento de placas de aço bruto. *Prêmio Mercosul de Ciência e Tecnologia*, 193–227.

Gupta, A., Mehta, A., Daver, L., and Banga, P. (2020). Implementation of storage in virtual private cloud using simple storage service on aws. *Second International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2020)*. doi:10.1109/icimia48430.2020.9074899.

Hussain, R.F., Pakravan, A., and Salehi1, M.A. (2020). Analyzing the performance of smart industry 4.0 applications on cloud computing systems. *IEEE International Conference on High Performance Computing and Communications (HPCC 2020)*. URL `\Url{https://arxiv.org/abs/2012.06054}`.

IEEE (1993). IEEE standards collection: Software engineering. IEEE Standard 610.12-1990.

indeed (2021). Salaries in the usa. URL `https://www.indeed.com/career/salarie`.

Jain, R., Kasturi, R., and Schunck, B. (1995). *Machine Vision*. McGraw-Hill.

Júnior, A.M.W. (2020). Estudo e aplicação de visão computacional e redes neurais para localização e detecção de falhas em montagem de pcbs.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). *Deep Learning*. Springer. DOI: http://dx.doi.org/10.18224/est.v35i2.644.

Lima, J.P.O., Oliveira, J.V.M., and Oliveira, R.C. (2019). Distributed artificial intelligence for recognition and position validation of printed circuit board. *European Academic Research*, 7, 4673–4689.

Lin, S. (2021). Research on automatic inspection system of printed circuit board based on computer vision. *Journal of Physics: Conference Series*. doi:10.1088/1742-6596/1861/1/012093.

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., García, A.L., Heredia, I., Malík, P., and Hluchý, L. (2019). Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, 52, 77–124. DOI: https://doi.org/10.1007/s10462-018-09679-z.

Schwab, K. (2017). *The Fourth Industrial Revolution*. Crown Business, 1st edition.

Singh, M.K. and Joshi, M. (2020). Cnn based automatic quality control system for pcb board defect detection using computer vision. *International Research Journal of Modernization in Engineering Technology and Science*, 2. doi:10.1088/1742-6596/1861/1/012093.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*. DOI: https://doi.org/10.1155/2018/7068349.

Zeng, L., Li, E., Zhou, Z., and Chen, X. (2019). Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial internet of things. *IEEE Network*, 33(5), 96–103. doi:10.1109/mnet.001.1800506.