INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* SABARÁ
SISTEMAS DE INFORMAÇÃO

Matheus Giovanny Oliveira Guimarães

REDES NEURAIS CONVOLUCIONAIS PARA RECONHECIMENTO FACIAL EM PHP:

avaliação de biblioteca php para reconhecimento facial

Sabará - MG
2024

MATHEUS GIOVANNY OLIVEIRA GUIMARÃES

REDES NEURAIS CONVOLUCIONAIS PARA RECONHECIMENTO FACIAL EM PHP:
avaliação de biblioteca php para reconhecimento facial

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação Ciência e Tecnologia de Minas Gerais - *Campus* Sabará para a obtenção do título de bacharelado em Sistemas de Informação.

Orientador: Prof. Dr. Carlos Alberto Severiano Junior

Sabará - MG
2024

Guimarães, Matheus Giovanny Oliveira

G963r Redes Neuras convolucionais para reconhecimento facial em PHP: avaliação de biblioteca PHP para reconhecimento facial [manuscrito]. / Matheus Giovanny Oliveira Guimarães. - 2024.

34 f. : il.

Orientação: Prof. Dr. Carlos Alberto Severiano Júnior.

Trabalho de Conclusão de Curso (Bacharelado de Sistemas de Informação) – Instituto Federal de Minas Gerais, *Campus* Sabará.

1. Redes neurais (Computação). – Monografia. 2. PHP (Linguagem de programação de computador). – Monografia. 3. Reconhecimento facial (Computação). – Monografia. 4. Estudos de viabilidade. – Monografia. I. Severiano Júnior, Carlos Alberto. II. Instituto Federal de Minas Gerais, *Campus* Sabará. III. Bacharelado de Sistemas de Informação. IV. Título.

CDU 004.8

César dos Santos Moreira / CRB6-2229
Biblioteca do IFMG *Campus* Sabará

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Sabará
Diretoria de Ensino, Pesquisa e Extensão
Conselho de Área - Informática e Comunicação
Rodovia MGC 262, Km 10 - Bairro Sobradinho - CEP 34590-390 - Sabará - MG
- www.ifmg.edu.br

ATA DE DEFESA DO TCC

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Aos doze dias do mês de novembro do ano de 2024, às quatorze horas, na Plataforma de Webconferência Google Meet iniciou-se a apresentação pública do Trabalho de Conclusão do Curso de Bacharelado em Sistemas de Informação, pelo discente **Matheus Giovanny Oliveira Guimarães**, intitulado **"REDES NEURAIS CONVOLUCIONAIS PARA RECONHECIMENTO FACIAL EM PHP: avaliação de biblioteca php para reconhecimento facial"**, tendo como orientador o Prof. Dr. Carlos Alberto Severiano Junior . O início dos trabalhos se deu com a apresentação da Banca Examinadora que foi composta pelos seguintes membros: Prof. Dr. Carlos Alberto Severiano Junior - Orientador, Prof. Me. Daniel Bruno Fernandes Conrado , Prof. Me. Luiz Guilherme Hilel Drumond Silveira, membros titulares. O discente iniciou sua apresentação, expondo seu trabalho durante trinta minutos. Os membros da banca apresentaram seus questionamentos e sugestões que foram respondidos pelo discente. A seguir, a Banca Examinadora reuniu-se, sem a presença do discente e do público, para fazer a avaliação final do trabalho apresentado. Em conclusão, a Banca Examinadora deliberou que o Trabalho de Conclusão de Curso foi:

(X) Aprovado.

( ) Aprovado com ressalvas.

( ) Reprovado.

Eu, Carlos Alberto Severiano Junior, Presidente da Banca Examinadora, lavrei a presente ata que será assinada por mim e pelos demais membros da Banca.

Sabará, 12 de novembro de 2024.

Documento assinado eletronicamente por **Carlos Alberto Severiano Junior**, **Professor**, em 12/11/2024, às 19:26, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Guilherme Hilel Drumond Silveira**, **Professor**, em 13/11/2024, às 11:49, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Daniel Conrado**, **Professor**, em 14/11/2024, às 13:16, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Matheus Giovanny Oliveira Guimarães

REDES NEURAIS CONVOLUCIONAIS PARA RECONHECIMENTO FACIAL EM PHP: avaliação de biblioteca php para reconhecimento facial

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação Ciência e Tecnologia de Minas Gerais - *Campus* Sabará para a obtenção do título de bacharelado em Sistemas de Informação.

Aprovado em: 12/ 11/ 2024 pela banca examinadora:

Prof. Dr. Carlos Alberto Severiano Junior - IFMG (Orientador)

Prof. Me. Luiz Guilherme Hilel Drumond Silveira - IFMG

Prof. Me. Daniel Bruno Fernandes Conrado - IFMG

Dedico esta monografia aos meus amados pais e irmãos, maiores incentivadores e fontes inesgotáveis de apoio, amor e compreensão.

AGRADECIMENTOS

Agradeço a toda à minha família, meus pais, meus irmãos e meus amigos por acreditarem em mim e pelo incentivo constante na realização deste trabalho.

Agradeço ao meu orientador e a todos que contribuíram de alguma forma para a realização deste trabalho.

“Todo o conhecimento genuíno tem origem na experiência direta."

Mao Tse-Tung

RESUMO

Este trabalho tem como objetivo comparar a implementação de redes neurais convolucionais utilizando PHP e comparada com Python, explorando suas capacidades no campo de reconhecimento facial. Em PHP, foi utilizada a biblioteca Rindow Neural Networks, enquanto em Python, a escolha foi pela biblioteca Keras com TensorFlow. Para a realização dos experimentos, ambas as linguagens seguiram estruturas semelhantes, utilizando a mesma base de dados de imagens distribuídas em 16 classes e sendo encapsuladas em frameworks web específicos, Slim para PHP e Flask para Python, além disso os códigos foram compilados na mesma máquina.

Os resultados indicam que a rede neural construída em PHP obteve uma acurácia média superior (0,954) em comparação com a rede em Python (0,86), embora o tempo de treinamento em PHP tenha sido maior. Todavia, o PHP demonstrou ser uma alternativa viável em contextos específicos, especialmente onde há uma forte dependência de tecnologias baseadas nessa linguagem, como projetos legado. No entanto, Python, com sua maior versatilidade e suporte da comunidade, ainda se destaca como a escolha preferida para projetos de aprendizado de máquina e para fins didáticos.

Embora o PHP possa ser utilizado para a construção de redes neurais, Python continua sendo a opção mais prática e eficiente na maioria dos cenários. Para análises futuras um estudo da biblioteca Rubix, outra ferramenta de aprendizado de máquina em PHP, pode ser feita para tentar superar as limitações identificadas pelo Rindow.

Palavras-chave: PHP. CNN. Viabilidade. Redes-Neurais

ABSTRACT

This paper goal is to compare an implementation of convolutional neural networks in both PHP and Python, expliciting their respectives performances in facial recognition. For PHP, the library Rindow Neural Networks was chosen for the project, meanwhile in Python, Keras using Tensorflow was the choice. For the experiments, both languages follow the same structure for the neural network, using the same dataset, compromised of 16 classes. Both codes were inserted in a web frameworks, Slim for PHP and Flask for Python.

The results show that the neural network built in PHP has a higher average accuracy (0,954) comparedd to the Python network (0,860), despite having a higher average training time. In the end, PHP proved to be a viable alternative in specific situations, namely when dealing with legacy projects and codebases. Despite that, Python with its bigger versatility and community support, is still the recommended option for machine learning projects and for learning enviroments, thanks to its ease of use and good documentation.

Despite the viability of PHP for building neural networks, Python is still the most practical and efficient choice for most scenarios. For future works an study of the Rubix library, another machine learning tool for PHP, could fix some of Rindow shortcomings.

Keywords: PHP. CNN. Feasibility. Neural-Networks

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A inteligência artificial, teve um aumento de popularidade graças ao lançamento do *ChatGPT* (RAY, 2023) no final do ano de 2022. Como um modelo de *chatbot* construído com redes neurais, o *ChatGPT* processa a resposta mais adequada para uma pergunta e aprende recursivamente com o *feedback* dos utilizadores. Desde então, outras companhias tentaram replicar o sucesso do modelo, criando por sua vez concorrentes como o *Gemini* (TEAM et al., 2023) e o *Claude 3* (ANTHROPIC, 2024). A corrida pelo modelo ideal trouxe avanços significativos para o campo de Inteligência Artificial e Aprendizado de Máquina, e sua aplicação já pode ser observada em outros aspectos da vida, como em reconhecimento de imagens (HU et al., 2015), sistemas bancários(LECUN et al., 1998) e casas inteligentes (ALMUSAED; YITMEN; ALMSSAD, 2023).

Já é possível ver tentativas de inserção de *IA* nos mais diversos cargos, inclusive desenvolvimento de software, com o protótipo recente do *Devin*, primeiro engenheiro de software totalmente artificial (COGNITION, 2024).

Na perspectiva de curto prazo, o uso dessas ferramentas vão se tornar cada vez mais comum, mas a subtituição total de postos de trabalho ainda é uma realidade distante (ACEMOGLU et al., 2022). No campo de desenvolvimento Web, as ferramentas de IA são majoritariamente utilizadas no apoio ao desenvolvimento, sendo predominantemente desenvolvidas em linguagens como Java e Python (VISIONX, 2024). O PHP, por sua vez, mesmo sendo uma linguagem popular, não é normalmente utilizado para o estudo desenvolvimento de aplicações *ML*. Existem alguns projetos que visam expandir o ecossistema da linguagem para a utilização de redes neurais, o mais popular sendo o *Rubix ML* (RUBIX, 2024) e o *Rindow Machine Learning* (ISHIKAWA, 2024a).

O PHP ainda é responsável por mais de 75 por cento dos sites webs em funcionamento em 2024 (TECHS, 2024) e é, constantemente uma das mais populares em desenvolvimento Web. Utilizar bibliotecas e ferramentas que permitem o uso de Machine Learning com a linguagem, pode ser uma vantagem importante em um time de especialistas em PHP, não necessitando de conhecimentos em outras linguagens para o desenvolvimento de MLs.

Neste trabalho, é feita uma comparação e análise a respeito da utilização de bibliotecas de Machine Learning em PHP, em relação a uma das linguagens mais populares nesse campo, o Python, e sua viabilidade para a produção de uma rede neural convolucional (ZHANG et al., 2023) para reconhecimento facial. Com o intuito de aprender e pesquisar sobre possíveis alternativas no próprio ecossistema PHP, garantindo assim uma menor curva de aprendizado e hipoteticamente até mesmo uma menor latência no uso da rede neural.

O objetivo deste trabalho é portanto, avaliar a viabilidade de utilização de uma biblioteca de aprendizado de máquinas em PHP em comparação com Python. Em específico a utilização do *Rindow Neural Networks* em PHP, comparado ao *Keras* em Python. Serão analisadas métricas

como acurácia, tempo de treinamento do modelo e tempo de predição. Além da viabilidade do Rindow para utilização em meio acadêmico e corporativo.

2 REVISÃO BIBLIOGRÁFICA

As redes neurais convolucionais são muito utilizadas para solucionar problemas de reconhecimento facial e de reconhecimento de imagem. Estudos (HU et al., 2015) mostram como a parametrização correta e seleção de camadas das redes escalona com o tamanho da base de dados analisados. A rede neural LeNet-5 modelada pelo professsor de computação Yann LeCun (LECUN et al., 1998) já provaram utilidade de redes neurais convolucionais para a identificação de cheques bancários nos Estados Unidos desde o começo do século XXI. Sua utilização comercial demonstra a eficácia das redes convolucionais em tarefas cotidianas. Apesar da LeNet-5 ter sido construída com o reconhecimento de digitos bancários em mente, sua estrutura inspirou a modelagem de redes de aprendizado profundo, principal tema de estudos recentes. Em (XIE; LI; SHI, 2019) a LeNet-5 volta a ser utilizada, em uma tentativa de adaptá-la ao reconhecimento facial, os resultados foram satisfatórios, demonstrando mais uma vez o êxito das redes neurais convolucionais em identificar imagens e padrões.

O êxito das redes neurais convolucionais para reconhecimento facial é explorado em inúmeros estudos acadêmicos, como o reconhecimento em ambientes irrestritos e com faces ocluídas(FREDJ; BOUGUEZZI; SOUANI, 2021). Nela é construído um algoritmo capaz de reconhecer faces até em ambientes que são de difícil identificação para o ser humano, como imagens escuras ou com pouca definição. Seguindo o êxito das CNN, a pesquisa sobre G-CNN e F-CNN (VINAY et al., 2017) se baseia em contruir modelos baseados em CNN que aprimoram ainda mais a acurácia de predição das faces em biometria, solidificando então a CNN como estado da arte para reconhecimento facial e reconhecimento de imagens na literatura acadêmica.

Em um campo mais amplo, o aprendizado profundo, no qual as redes neurais convencionais pertencem, são implementadas nas mais diversas linguagens. Destacando o Deeplearning4j (LEARNING4J, 2024) em Java, que permite o treinamento de modelos em Java, com interoperações com sistemas Python, com destaque a importação de modelos Keras. Permitindo a execução dos modelos sem necessidade de retreinamento. Com uma documentação profunda e mantida pela comunidade o Deeplearning4j é alternativa robusta para aprendizado profundo.

Tesmo também o (ABADI et al., 2016) em Python, que além de ter inúmeras funcionalidades, permite o uso de sua API para integração com os mais diversos sistemas. Sendo assim, o *TensorFlow* é uma parte importante do ecossistema de aprendizado de máquina em Python, sua interface API sendo inclusive utilizada pelo Keras.

Em C++ temos o *Caffe* (JIA et al., 2014). Destacando-se pelo sua modularidade e velocidade, redes neurais em *Caffee*, conseguem processar mais de 60M de imagens por dia, em aproximadamente 5/milissegundos divididos entre aprendizado e inferência para cada imagem. Logo as redes neurais são possíveis de serem implementadas em qualquer linguagem de programação contanto que elas sejam Turing-completas.

3 REFERENCIAL TEÓRICO

O aprendizado de máquina, em inglês, *Machine Learning* é uma forma de algoritmo que é capaz de aprender e conceder soluções a problemas, a partir de um *dataset* fornecido (GOODFELLOW; BENGIO; COURVILLE, 2016). O aprendizado profundo ou *Deep Learning*, por sua vez é uma rede neural em que o algoritmo de múltiplas camadas, extrai as informações necessárias para a base de dados por si mesmo e aprende sobre o problema, retirando ainda mais a participação humana no processo (ZHANG et al., 2023). A utilização de diversas camadas permite que os algoritmos de aprendizado profundo aprendam padrões complexos, como imagens, vídeos e aúdio.

Primeiro devemos entender o que é um algoritmo de aprendizado de máquina e como ele se difere estruturalmente de um algoritmo de *Deep Learning*, aprendendo sobre a parametrização da rede neural, a construção de suas camadas, funções e estrutura.

3.1 Aprendizado de Máquina

Um algoritmo de aprendizado de máquina é aquele que consegue aprender a partir da própria experiência observacional, quanto mais expêriencia um algoritmo acumula, mais eficiente ele se torna .O que significa que, em vez de ser explicitamente programado para realizar uma tarefa específica, ele consegue, ao observar dados e experiências anteriores, melhorar sua capacidade de realizar essa tarefa. Em exemplos práticos, este seria um algoritmo que observaria problemas onde não há um padrão definido e a partir da observação dos dados e da ativação de funções algorítmicas, criaria soluções para o problema (ZHANG et al., 2023). Isso acontece através da adaptação de seus parâmetros internos, o que permite que ele reconheça padrões, mesmo em problemas complexos onde não há um padrão definido ou óbvio. Essa solução por si, é avaliada pelo parâmetro definido como alvo para a rede neural assim como outras métricas como, vieses ou peso. Esse processo de aprendizado ocorre por meio de diversos modelos, tais como as redes neurais artificiais, que são inspiradas na estrutura dos neurônios do cérebro humano ou as redes neurais convolucionais, inspiradas na visão humana.

3.1.1 Modelo

Um modelo de aprendizado é um algoritmo que é utilizado para identificação de padrões e resoluções de problemas. O modelo é projetado para aprender com a base de dados fornecida, ele inclui os chamados parâmetros e hiperparâmetros, configurações que modificam o processo de aprendizado (GOODFELLOW; BENGIO; COURVILLE, 2016).

O modelo pode ser treinado, um processo que ensina ao algoritmo a realizar predições ou decisões baseado no conjunto fornecido, o ajuste fino de parâmetros durante o processo de treinamento é o que melhora a sua performance de acordo com as métricas selecionadas.

3.1.2 Base de dados

A base de dados ou *Dataset* é a informação necessária para o processamento do modelo, a partir dele é possível treinar e testar o algoritmo. Tipicamente inclui os conceitos de *inputs* e *labels* usados para classificar as classes do dataset e treinar o algoritmo no processo real (ZHANG et al., 2023). Antes de possuir um uso real para o modelo, o *dataset* pode então ser preprocessado antes, isto inclui a normalização, limpeza e transformação das informações, após transformada é dividida entre dois conjuntos:

- **Treinamento**: o conjunto que será utilizado para treinar o modelo, normalmente utiliza-se 80% do *dataset* original.
- **Teste**: o conjunto que será utilizado para testar o modelo após o seu treinamento, utilizado para validar a performance dele, normalmente utiliza-se 20% do *dataset* original.

3.1.3 Algoritmo de Aprendizado

Quando temos um modelo definido e as informações obtidas, cabe decidir como procurar a solução ideal para o problema. Como dito anteriormente, o aprendizado de máquina aprende com a própria experiência e aqui entram os algoritmos de aprendizados (ZHANG et al., 2023). Nos algoritmos é importante também a implementação de uma função de perda, uma função responsável por identificar a discrepância entre os valores previstos pela rede neural, e os valores reais presentes no dataset de teste, algumas de suas formas são o chamado erro quadrático médio e a entropia cruzada (ZHANG et al., 2023). Afim de alinhar os dados de perda previstos com os reais e melhorar a performance do algoritmo, utiliza-se os otimizadores, responsáveis por minimizar ou maximizar a função de perda iterativamente até atingir uma performance aceitável, algumas de suas implementações são o método do gradiente estocástico e a função ADAM (*Adaptive Moment Estimation*) (ZHANG et al., 2023).

3.1.4 Métricas de avaliação e testagem

Após o treino utlizando o algoritmo, a função de perda e os otmizadores escolhidos, chega a hora de avaliar a efetividade do modelo, para isso são utilizadas as métricas de avaliação, e elas variam de acordo com a tarefa proposta pela modelo inicialmente, é comum utilizar-se da acurácia, precisão e o *recall* para se avaliar a tarefa.

Por fim, existe a testagem e validação do modelo, na seção 3.1.2 vimos que uma porção do dataset é separado para realizar a testagem do modelo, esta é utilizada para testar o modelo em um dataset não visto antes, visto que ele só foi treinado com parte do dataset original.

O propósito de avaliar e testar um modelo é ver se ele é adequado para solução de dado problema, especialmente se comparado a outros modelos tentando solucionar a mesma tarefa.

3.2 Aprendizado Profundo

Nas décadas recentes os avanços no campo de processadores e principalmente de placas gráficas, permitiram o processamento de *datasets* cada vez maiores, quantidades de dados que seriam incalculáveis a décadas atrás. As *GPU*, são importantes para o aprendizado profundo por permitirem o processamento paralelo de dados em grandes quantidades. Graças ao seus milhares de núcleos, além da otimização delas para operações de matriz e de algebra linear. Em contrapartida, tiveram poucos avanços na capacidade de mémoria de acesso aleatório (*RAM*), que por sua vez são importantes para garantir a manipulação de modelos grandes. Dessa forma os algoritmos deveriam ser cada vez eficientes a fim de processar a grande quantidade de dados (ZHANG et al., 2023). Foi então que os algoritmos de aprendizado profundo ou *Deep Learning* em inglês foram popularizados (LECUN et al., 1998).

Em um exemplo prático, suponhamos que um programador foi designado com a função de criar um algoritmo que identifique a partir de uma foto, presente vários rostos e objetos, o nome e o contorno de cada rosto e de cada objeto. Tal tarefa requer um nível de processamento que vai além de um mero algoritmo linear de aprendizado de máquina, dissecando a imagem, existem inúmeras combinações de pixels que podem ser identificados como um objeto ou pessoa. A Figura 1 é uma figura retirada de um site.

Figura 1 – Ilustração de uma visão computacional.

Fonte: (SCIENCE, 2024)

Para resolver este problema em um algoritmo de aprendizado de máquina linear, seria necessário a extração manual dos atributos que tornam cada elemento da imagem único, o seu formato, intensidade de cor ou o seu tamanho. A efetividade do algoritmo, dependeria então

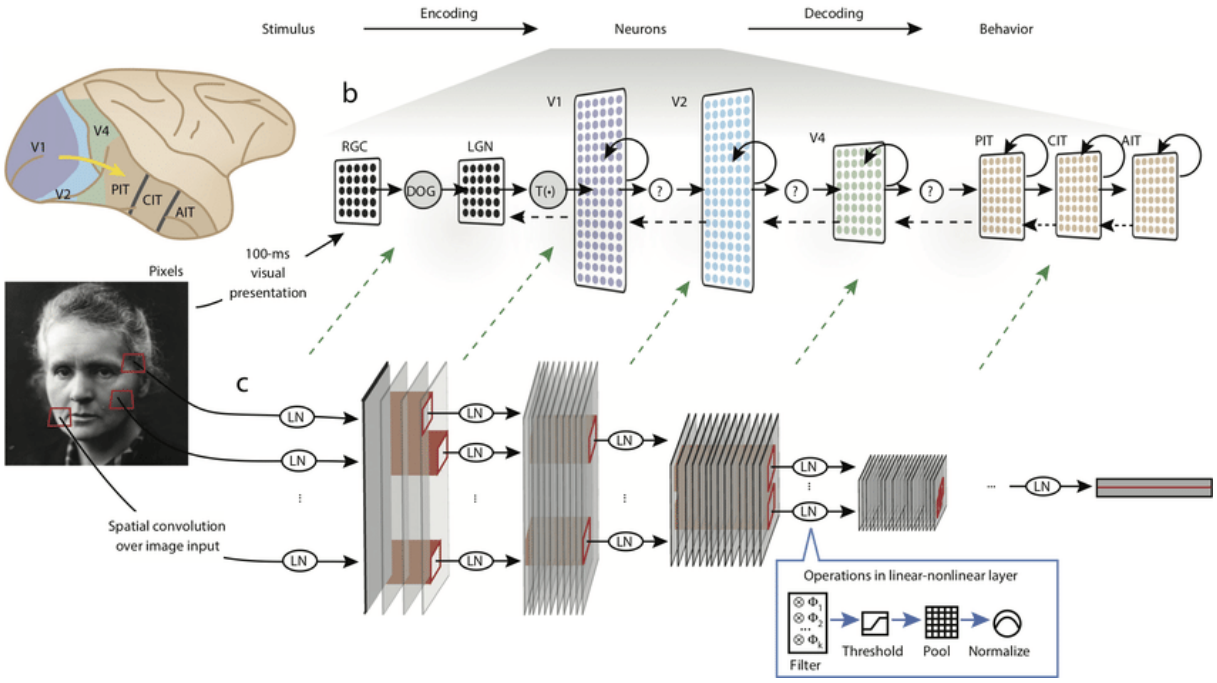
fortemente do quão bem selecionado foram os atributos pelo pesquisador. Em um algoritmo de aprendizado profundo, não é necessária a extração dos atributos manualmente, o modelo aprende sozinho a selecionar, isolar e classificar os atributos que definem um elemento. Isto é feito através de múltiplas camadas de processamento (ZHANG et al., 2023), que identificam caracteristicas que podem classificar a entrada, como por exemplo a presença de arestas ou bordas no caso de uma imagem para visão computacional. A medida que o algoritmo aprende essas características, elas são combinadas oque o torna capaz de identificar padrões cada vez mais complexos.

Pela dificuldade de categorização de vários atributos em dezenas, talvez centenas de elementos, o algoritmo de aprendizado profundo se torna mais adequado para tratar de grandes bases de dados.

3.2.1 Redes Neurais

São um modelo de aprendizado de máquina, inspirados pelo cerébro humano, mais especificamente a sua estrutura de neurônios. Uma rede neural possui uma camada de entrada de informações que recebe os dados brutos para serem processados. Seguido de camadas ocultas que realizam o processamento destes dados, em características que vão auxiliar na classificação e finalmente as camadas de saídas que vão calcular, prever ou classificar dado inserido.

Figura 2 – Ilustração de uma rede neural, na parte superior o processo de pensamento humano, na parte inferior uma rede neural inspirada neste processo.



Fonte: (YANG, 2016)

Na figura 2 é possível observar o processo descrito e a similaridade da rede neural com o processo de formação de pensamento humano. Com enfâse nas etapas de codificação

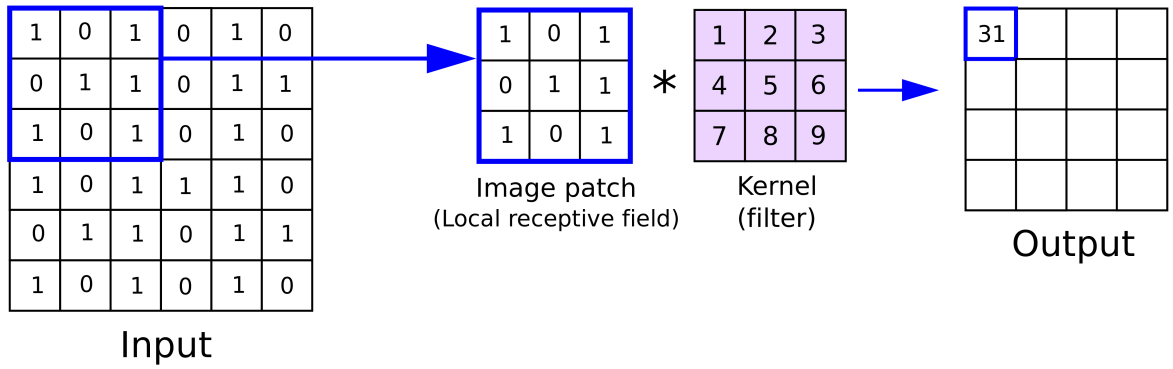
das características que podem ser úteis na classificação. No exemplo da figura, arestas, bordas ou diferenças de contrastes podem significar a presença de uma olho, de uma orelha ou de cabelo. A junção dessas características permitem a identificação de um rosto humano. Na figura o processo de classificação pertence especificamente à uma rede neural convolucional, mas a mesma definição de processo pode ser aplicada a outros tipos de redes neurais.

3.2.2 Redes Neurais Convolucionais

Em inglês *Convolutional Neural Networks* (CNN) (GOODFELLOW; BENGIO; COURVILLE, 2016), são uma forma especializada de aprendizado profundo ideal para identificação de imagens e vídeos. Inspiradas pela visão humana, sua arquitetura única de multicamadas permite a análise e identificação de padrões em imagens das mais diferentes complexidades.

A fundação de um modelo CNN é a camada de convolução, nela a rede extrai as características únicas da imagem utilizando um filtro (ou *kernel*), esses filtros são pequenas matrizes que percorrem a imagem realizando o produto matricial entre a região escaneada e os seus valores. Este processo é chamado de convolução, a medida em que o filtro se move pela imagem, uma matriz de atributos é calculada e a partir dela a atráves da sopreposição dessas matrizes a rede consegue detectar possiveís arestas, cores ou texturas (GOODFELLOW; BENGIO; COURVILLE, 2016).

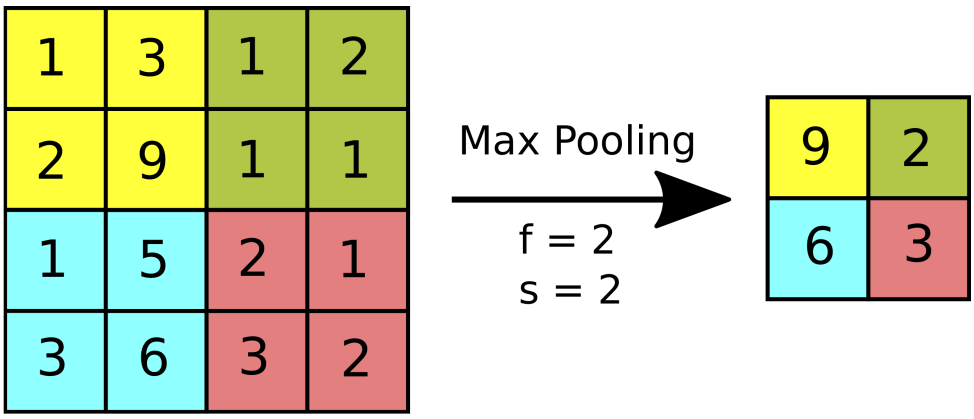
Figura 3 – Ilustração do processo de convolução.



Fonte: (POOLING, 2019)

Após a camada de convolução temos a camada de *pooling*, nela são reduzidas as proporcionalidades dos atributos detectados afim de reduzir a carga computacional na rede, sem distorcer as informações relevantes. Um exemplo de operação é a *max pooling* (ZHANG et al., 2023).

Figura 4 – Ilustração do processo de *pooling*.



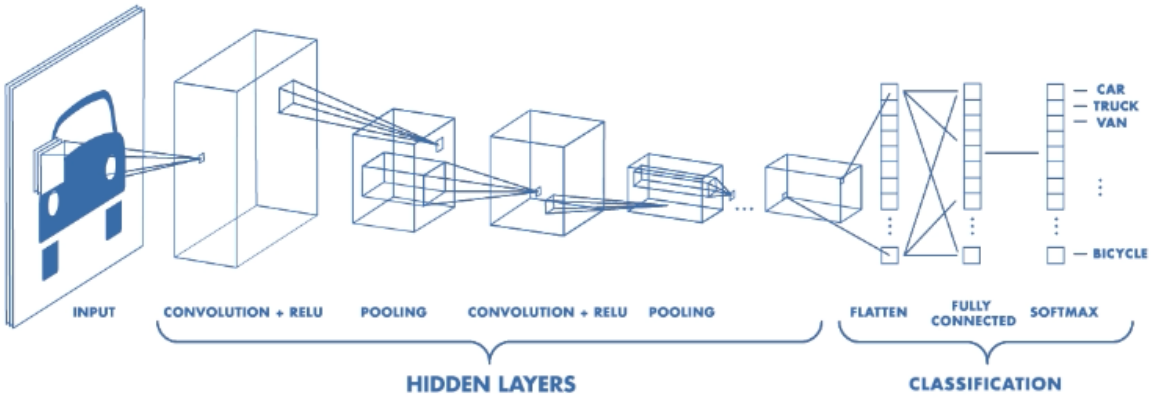
Fonte: (**??**)

Note que as camadas podem ser estendidas quantas vezes foram julgadas necessárias, é comum vermos padrão de CNNs da seguinte maneira:

Convolução → Pooling → Convolução → Pooling → ⋯ → Camada totalmente conectada

A seguir os dados passam pelo processo de *flattening* em caso de estruturas de dados multidimensionais e finalmente são conectadas as camadas de processamento da rede neural. Na figura 5 temos uma representação completa desse processo.

Figura 5 – Ilustração de uma CNN completa.



Fonte: (MATHWORKS, 2017)

Em resumo as CNNs são poderosas redes neurais, muito comumente utilizadas na detecção de padrões em imagens (XIE; LI; SHI, 2019) (LECUN et al., 1998). Por este motivo, ela foi selecionada como sendo parte fundamental do experimento deste trabalho.

3.3 Bibliotecas de reconhecimento facial

Com a finalidade de facilitar o desenvolvimento de redes neurais, uma série de bibliotecas estão disponíveis para que programadores possam se aprofundar no aprendizado ou desenvolver aplicações e soluções. Entre algumas das mais populares, estão os projetos de código aberto Tensorflow e Keras para serem utilizadas com a linguagem Python, mas no ecossistema da linguagem existem outras bibliotecas utilizadas para no aprendizado de máquina, como Scikit-learn e o Pytorch. Em PHP, temos poucas opções de bibliotecas, a mais popular sendo a Rubix surgida em 2018 com o intuito de reativar o desenvolvimento de redes neurais em PHP após o deseaceleramento e eventual descontinuação do PHP-ML. Formalmente a iniciativa mais avançada em aprendizado de máquina em PHP, contamos também com o Rindow Machine Learning, especializada em redes neurais convencionais.

3.3.1 Bibliotecas em Python

Criada em 2015, o Tensorflow (ABADI et al., 2016) é uma biblioteca de aprendizado de máquina de código aberto desenvolvido pela Google que rapidamente dominou o campo acadêmico devido a sua facilidade de uso e flexibilidade. Em seus 9 anos de funcionamento, foi desenvolvida uma comunidade e ramificações do projeto que o tornaram um dos mais robustos no campo,

O Keras (KETKAR; KETKAR, 2017), também desenvolvido pela Google, foi lançado também em 2015, com o objetivo que criar uma interface API que facilitaria ainda mais a criação de modelos neurais. Ele fornece uma interface de modelagem que é baseada em alguma biblioteca de aprendizado de máquina e facilita a criação dos modelos, sendo usada em conjunto com o Tensorflow ou outras bibliotecas.

Devido a facilidade de uso e popularidade, foi escolhida ambos Tensorflow e Keras para o desenvolvimento de uma CNN deste trabalho, utilizando-se de recursos da comunidade e a documentação rica de ambos.

3.3.2 Bibliotecas em PHP

Em PHP, as opções de bibliotecas para desenvolvimento de aprendizado de máquinas são bem mais limitadas, por muitos anos a opção mais popular foi a PHP-ML (KONDAS, 2022), e permitia a a criação de modelos de redes neurais utilizando completamente o PHP, em meados de 2020 o projeto foi desativado e sua pagina no Github apagada, devido a um conflito autoral.

O Rubix ML surgiu em 2018 adjacente ao desenvolvimento do PHP-ML, a proposta do Rubix era permitir a implementação de redes neurais compatíveis com a versão mais recente, PHP8, nesta versão considerada um marco para a linguagem, a performance foi significantemente aprimorada, graças a implementação da compilação JIT (Just In Time) na linguagem. Após a

descontinuação do PHP-ML em 2020, o Rubix se tornou a principal biblioteca de aprendizado de máquinas e hoje conta com uma comunidade engajada.

Apesar de ser uma biblioteca bem robusta e em constante desenvolvimento, o Rubix possui suas limitações, no atual momento não permite o desenvolvimento de redes neurais convolucionais. É neste contexto que utiliza-se o Rindow Machine Learning, uma biblioteca surgida em 2020 e que se especializa em redes neurais convolucionais em PHP.

3.4 PHP e aprendizado de máquina

No momento em que este trabalho foi desenvolvido, não foram encontrados em português ou inglês, nenhum trabalho acadêmico relevante sobre aprendizado de máquina com PHP, porém ela é uma linguagem de programação que recebe novas capacidades de processamento a cada versão, incluindo o *JIT* na versão 8. Um dos desenvolvimentos mais recentes para a linguagem foi a criação do NumPower (NUMPOWER, 2024). Inspirado pelo NumPy a biblioteca fornece base para matemática computacional com PHP, a equipe de desenvolvimento do Rubix já está em processo de integração do NumPower para o projeto, mas não há novidades a respeito do Rindow e a adoção da biblioteca. As expectativas para a biblioteca são que ela ajude a impulsionar não só as bibliotecas de *Machine Learning* em PHP, como também incentivar a matemática computacional e científica com a linguagem.

O interesse do desenvolvimento de aprendizado de máquina com PHP é recente, apesar da linguagem ser perfeitamente capaz de realizar as operações nécessárias para *Machine Learning*. Dentre os maiores obstáculos para o desenvolvimento com a linguagem, estava o desempenho inferior se comparada a outras linguagens mais usuais para aprendizado de máquina, mas este ponto já foi solucionado como dito previamente. A inabilidade de aceleramento por placa gráfica, a *GPU* também limitava o escopo que as redes em PHP poderiam ter. Parte do pacote Rindow, está incluso a execução do OpenCL (interface que permite que o mesmo programa seja executado em diferentes plataformas, como na CPU e na GPU) como um plugin para o PHP, permitindo a aceleração das redes por *GPU*.

4 EXPERIMENTOS E RESULTADOS

Com o objetivo de comparar uma rede neural em PHP, com linguagens mais comumente utilizadas para aprendizado de máquinas, a segunda linguagem escolhida para a comparação foi Python, popular pela sua simplicidade, usabilidade e velocidade, com a biblioteca de aprendizado de máquina, Keras (CHOLLET et al., 2015). Em PHP optou-se pela utilização da biblioteca Rindow Neural Networks, por ser a única encontrada durantes as pesquisas que é capaz de modelar redes neurais convolucionais. As ferramentas utilizadas no experimento foram:

1. Código construído em PHP
    a) Rede neural construída com a biblioteca Rindow Neural Networks
    b) Matématica computacional com Rindow Mathematics
    c) Código encapsulado no micro framework Slim PHP
2. Código construído em Python
    a) Rede neural construída com um backend baseado na biblioteca Tensorflow
    b) Interface neural construída com a biblioteca Keras
    c) Código encapsulado no microframework Flask

Especificações do computador utilizado no experimento:

1. Processador: Ryzen 5 5600x, 6 cores, 12 threads, 4.6ghz max clock
2. Memória RAM: 16gb 3200mhz
3. Placa de vídeo: GIGABYTE GAMING GeForce RTX 3070
4. Placa mãe: Asus PRIME B450M-GAMING/BR

O tipo de rede neural escolhida foi uma rede neural convolucional, mais adequada para aprendizado com base de dados baseados em imagens, propósito inicial do experimento.

O dataset utilizado no experimento foi fornecido pelo artigo (HASHMI, 2021), e consiste de 308 imagens distribuídas igualmente entre 16 classes, separados por rostos. Em ambos os códigos, o experimento foi encapsulado em um *built-in webserver*, baseado no seus respectivos *frameworks*, *Flask* para o código baseado em Python e Slim para o PHP, com limite de memória de script ilimitada em ambos os casos. Devido ao fato de o PHP ser uma linguagem processada por *web servers*, houve a necessidade de se utilizar *frameworks web* para ambos os códigos, afim de trazer paridade de processamento entre as duas linguagens.

Caso não esteja claro, todas as atividades descritas na sessão foram realizadas em ambos os códigos.

4.1 Preparação da rede e dos dados

Em PHP, os dados foram preparados utilizando as classes da biblioteca Rindow NN, a classe ImageDataGenerator permite a prepação dos dados, selecionando parâmetros como, proporções e qualidade da imagem. Em Python os dados foram preparados utilizando os *dataset generators* da biblioteca Keras. Foi separada a base de dados entre treinamento e teste contendo respectivamente, 244 e 64 imagens, foi normalizada a fim de ter ambas altura e largura de 64 pixels, elas são então embaralhadas no gerador para então poderem ser utilizadas no treinamento do modelo.

A rede pôde ser construída da mesma maneira em ambas as linguagens, os nomes de métodos e parâmetros são quase que intercambiáveis entre as duas bibliotecas, permitindo então que um especialista em aprendizado de máquina, possa facilmente migrar entre as duas linguagens e facilitando a compreensão de projetos. Para maiores detalhes, o código fonte de ambos os projetos em PHP (GUIMARAES, 2024a) e Python (GUIMARAES, 2024b) podem ser acessados atráves da plataforma Github.

A diferença central portanto, é uma questão de familiaridade com a linguagem, já que os métodos e camadas possuem uma estrutura e lógica similar entre as duas. Uma visualização da rede construída nas duas linguagens é apresentada na tabela 1:

Camada	Formato de Saída	Parâmetros
Conv2D	(60,60,32)	2432
MaxPooling2D	(30,30,32)	0
Conv2D	(26,26,64)	51264
MaxPooling2D	(13,13,64)	0
Flatten	(10816)	0
Dense	(64)	692288
Dense	(16)	1040

Tabela 1 – Descrição das camadas da rede neural em ordem de implementação.

A rede é iniciada com uma camada de *Conv2D*, que é responsável por fazer o processo de convolução da imagem através de um filtro, no começo da rede, de tamanho 32. Cada filtro vai gerar um mapa de características específicos para a imagem, oque auxilia no processo de identificação de características.

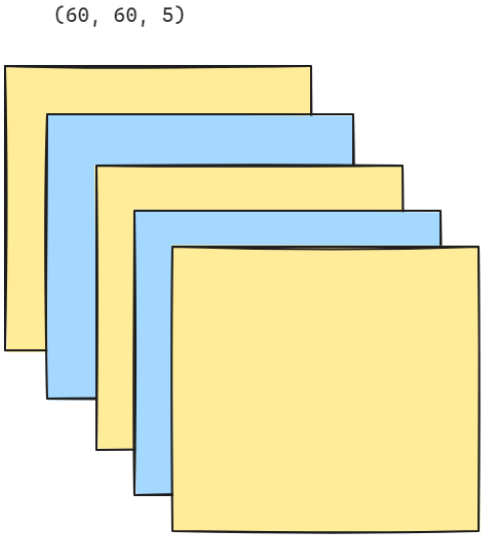
Em seguida temos uma camada de *MaxPooling2D*, essa camada reduz a dimensionalidade do mapa de características gerado pela camada *Conv2D*, diminuindo a sua resolução proporcionalmente. Sem perda de definição de características.

A camada de *Flatten* converte o mapa de características que é um vetor em 3 dimensões, uma para cada um dos canais RGB, em um vetor de uma única dimensão. Isto é feito para que ele possa ser passado para a camada seguinte, que realizará a classificação das características.

Por último, na camada de *Dense* totalmente conectada, ou seja, onde cada neurônio é conectado a todos os neurônios da camada anterior, é usada para tomar as decisões e fazer as classificações com base nas características extraídas das camadas anteriores.

O Formato de saída são as proporções as camadas na saída, normalmente representada pela notação de três dígitos: (X, Y, Z), onde X é altura, Y é largura, Z são os diferentes mapas de características obtidos após aplicar o mesmo número de filtros. Na rede é possível observar as proporções decrescentes da imagem a cada aplicação da camada de *MaxPooling2D*. A representação de um formato de saída hipotético pode ser visto na figura 6.

Figura 6 – Representação visual de uma camada hipotética $(60, 60, 5)$.



Fonte: Própria do autor

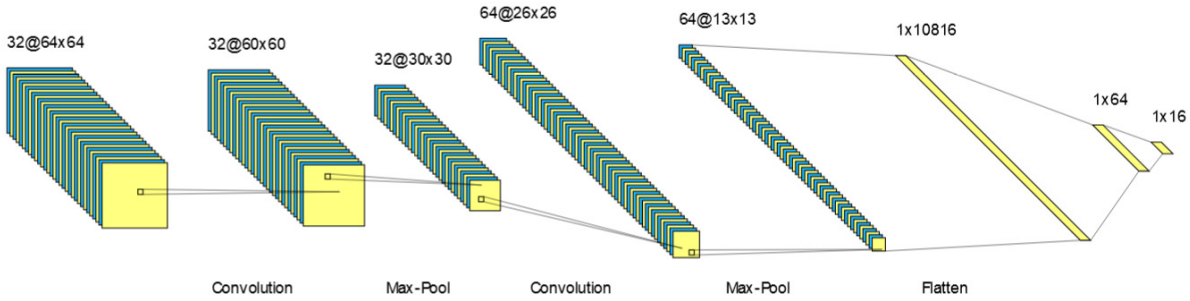
Os parâmetros indicam a quantidade de pesos e vieses que o modelo precisa treinar naquele momento, em cada camada a quantidade de parâmetros e dado de formas diferentes como observado abaixo:

- **Conv2D (2432 e 51264)**: Em uma camada Conv2D, o número de parâmetros é dado pelo número de filtros multiplicado pelo tamanho do kernel (5x5) e acrescentado de um valor de viés para cada filtro.
- **MaxPooling2D (0)**: As camadas de MaxPooling não têm parâmetros treináveis, pois só realizam uma operação de redução.
- **Flatten (0)**: Também não possui parâmetros treináveis, pois apenas modifica a forma dos dados.

- **Dense (692288 e 1040)**: Em camadas totalmente conectadas, responsáveis pela classificação, os parâmetros são a multiplicação do número de neurônios na camada atual pelo número de entradas das camadas anteriores, adicionado a um viés para cada neurônio.

Portanto a rede final pode ser visualizada na imagem a seguir:

Figura 7 – Representação visual da rede neural construída .



Fonte: Própria do autor

Foram definidas rotas web com o intuito de permitir a execução de diferentes operações de análise treinamento e predição, atráves do servidor. São as seguintes:

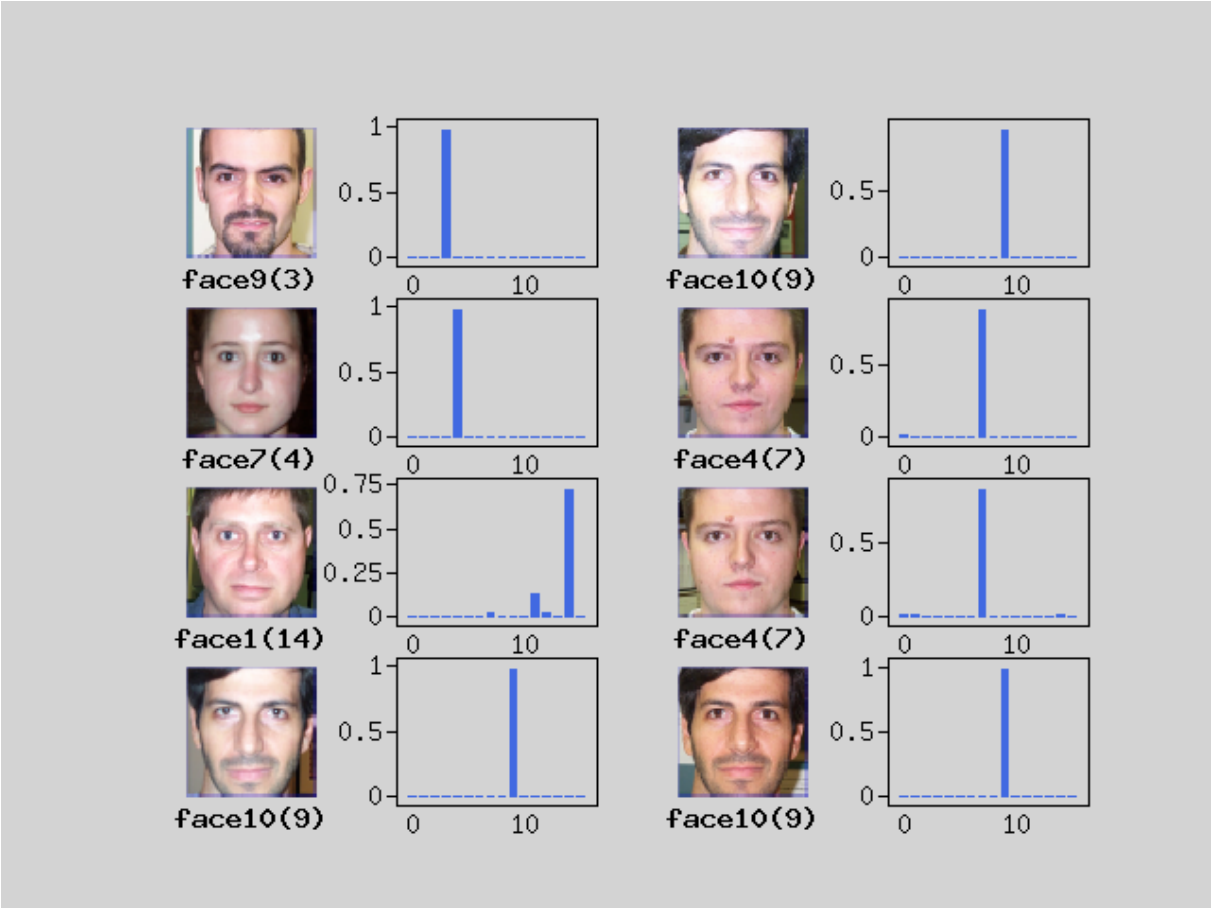
- **/train**: treinamento do modelo, os dados são preparados e o modelo treinado, e salvo como um arquivo *.model* em PHP e como um *.h5* em Python, são exibidas as camadas da rede neural, a acurácia e o tempo de treinamento do modelo.
- **/train/stress**: realiza o treinamento do modelo 30 vezes, é então feita a média da acurácia e do tempo de todos os treinamentos, as médias são exbidas no navegador.
- **/predict/image/<image>**: realiza a predição da imagem selecionada no argumento $<$ *image* $>$, exibe o resultado da predição no navegador.
- **/predict/plot**: realiza a predição de uma imagem aleatória, exibe o resultado da predição como um gráfico de barras, rota exclusiva do código em PHP.
- **/predict/batch**: realiza a predição de 8 imagens aleatórios a partir da base de dados de teste, o resultado é exibido em gráfico de barras, rota exclusiva do código em PHP.

Somente as rotas de **/plot /batch** exibem imagens no navegador, todas as outras rotas exibem os valores calculados no <body> da pagína, sem nenhum tipo de estilização.

4.2 Teste e predição

Para a a testagem do modelo foram utilizada 4 imagens de cada classe, totalizando 64 imagens no *dataset* de teste, foram executados 10 *epochs* para o *fit* da rede. As métricas observadas foram o tempo de treino e a acurácia do modelo.

Figura 8 – Representação gráfica da predição de 8 faces aleatórias.



Fonte: Própria do autor

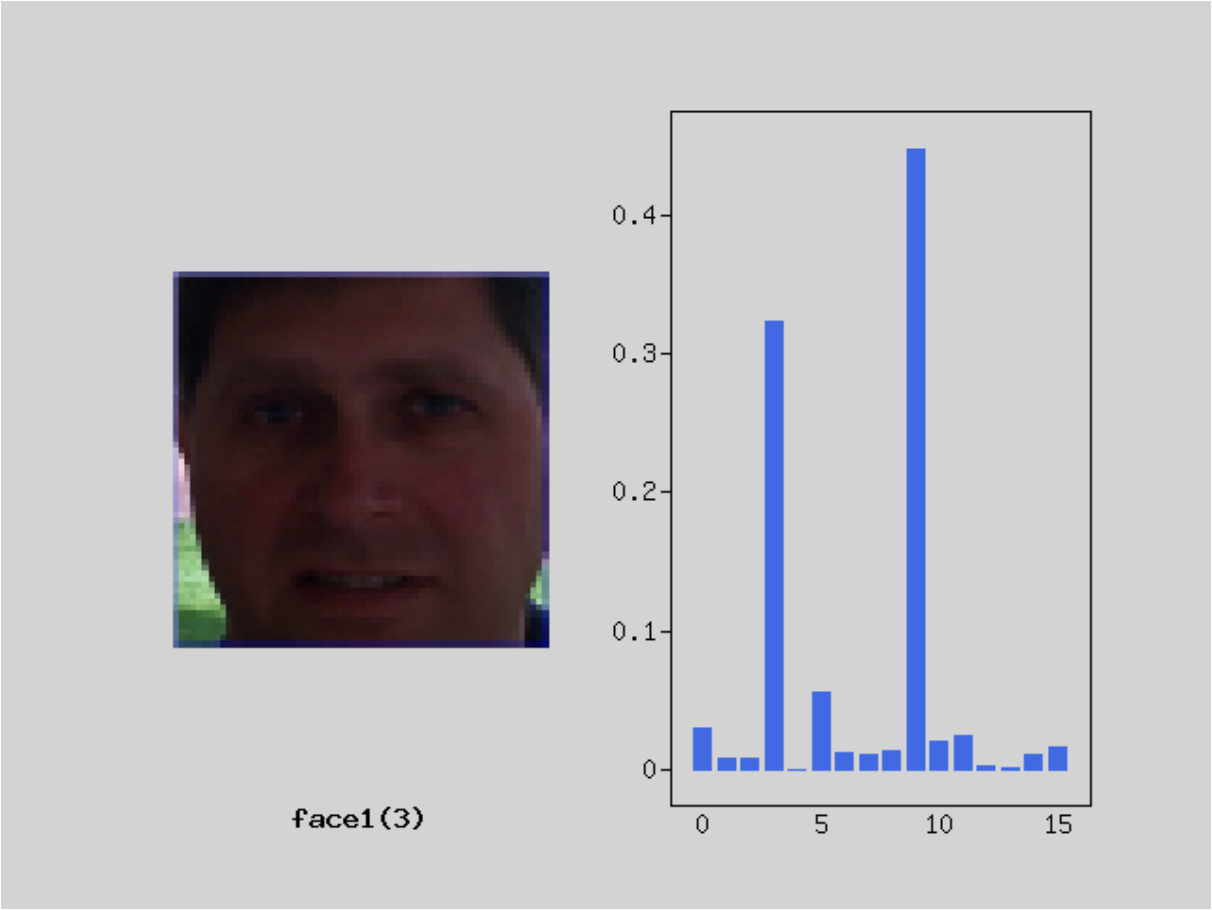
Para medir o tempo médio de execução do treinamento da rede neural, foram executados 30 treinamentos em sequencia e os resultados de tempo e acuracia foram armazenados. O experimento encontrou os seguintes resultados presentes na tabela 2.

Rede Neural	**Tempo médio para treinamento**	**Acurácia média do modelo**	**Tempo médio de predição**
PHP	45.45 segundos	0.954	11.542 milissegundos
Python	7.15 segundos	0.86	12.541 milissegundos

Tabela 2 – Métricas de treinamento e predição das redes neurais.

O desempenho do código em PHP foi significantemente maior no quêsito acurácia, compensado porém por um tempo de treino maior para o modelo, mas para casos de uso reais, a predição de uma imagem ou base de dados é muito mais relevante para análise. Neste ponto ambas as redes tiverem tempos de predição em milissegundos em média de 12ms e portanto são irrelevantes para a análise. Em casos pontuais, imagens que não identificadas pelo visão humana, também não são identificadas pelas redes neurais. Observe abaixo em das imagens da face1.

Figura 9 – Imagem pertencente a face1, com indíce 3.



Fonte: Própria do autor

A predição correta para esta imagem da face1 seria a posição [3] no vetor de predição, já que este índice pertence a face1, a predição porém encontrou o índice [9] como o mais próvavel da imagem pertencer, índice que pertence a outra face.

4.2.1 Análise de viabilidade

Em termos de acurácia absoluta nas predições, a rede neural em PHP lidera com uma vantagem significativa, apesar do seu treinamento mais longo, mas considerando que em um cenário real, o treinamento do modelo só é realizado uma única vez, o tempo de treinamento se torna um fator menos valioso. O Python por sua vez, possui uma vantagem significante

sobre o PHP, por ser uma linguagem de uso geral e não exclusiva para desenvolvimento web, a possibilidade de compilar um código Python sem utilizar um web server, o torna mais versátil para utilização nos mais diversos projetos. Outro fator a se considerar é o suporte e documentação que as bibliotecas de redes neurais possuem, neste quesito o Keras, biblioteca utilizada para a rede em Python, não só possui uma documentação robusta, como também uma comunidade maior e mais engajada, portanto a colaboração mútua em projetos de código aberto é mais fácil. Já a biblioteca Rindow Neural Networks, possui uma documentação menos completa e confusa, motivo este que se deve ao idealizador do projeto não ser um falante nativo de inglês, logo a tradução da documentação foi feita inteiramente por tradução de aprendizado de máquina (ISHIKAWA, 2024b), além disto, a biblioteca possui uma comunidade pequena, com poucos projetos colaborativos. No quesito acessibilidade de uso da biblioteca, ambas são fáceis de se utilizar. Em PHP todas as dependências podem ser geridas pelo *Composer*, gerenciador de bibliotecas mais comum para linguagem, já em Python, todas as bibliotecas podem ser geridas pelo PIP, padrão para a linguagem, logo a facilidade de uso se torna uma questão de familiaridade com a linguagem.

Na tabela 3 abaixo, podemos ver uma compilação de outras métricas para a avaliação das redes:

Métrica	Rindow Neural Networks (PHP)	Keras (Python)
Aceleramento por GPU	Sim	Sim
Compilação	Via web-server	Nativo
Documentação	Incompleta	Completa
Facilidade de uso	Fácil	Fácil

Tabela 3 – Métricas de avaliação de viabilidade de uso das redes neurais.

Apesar do desempenho de acurácia maior fornecido pelo Rindow, a falta de suporte do pacote e a inconveniência da compilação do código em PHP, torna uma rede neural nesta linguagem menos atrativo para casos reais, onde um código que possa ser compartimentalizado e compilado nativamente são mais úteis. Porém, para um time de especialistas em PHP, o Rindow pode ser uma ferramente poderosa para criações de modelos neurais sem necessitar da utilização de outras linguagens de programação, assumindo-se que a estrutura necessária para a compilação do PHP em server já esteja preparada neste caso. Considerando que mais de 75 por cento dos sites webs em funcionamente estão escritos em PHP e quase 14 por cento destes em versões inferiores ao PHP 5 (TECHS, 2024), nestes casos de projetos legado de difícil manutenção e principalmente de integração com outras ferramentas, o Rindow abre portas para a implementação de redes neurais sem a necessidade de refatoração e riscos advindos do processo.

Para fins de didáticos, em aulas, cursos e workshops, o Python com Keras é neste caso, ainda uma escolha mais adequada, não só por ser uma linguagem de sintaxe simples, mas por

possuir um suporte de ecossistema muito mais favorável para aprendizado de máquina. Além de ferramentas como Jupyter Notebook e Google Collab que facilitam o ensino e a colaboração.

5 CONCLUSÃO E TRABALHOS FUTUROS

Este estudo analisou a viabilidade de uma rede neural construída em PHP, com uma construída em uma das linguagens mais comuns para este fim, o Python. Explicitando seus casos de uso, vantagens e desvantagens. A partir dos resultados coletados pela medição das métricas de acurácia, tempo de predição de imagem e tempo de treinamento, que foram obtidos após a medição de 30 modelos gerados. Com isso foi possível concluir que, apesar do Rindow ter obtidos uma acurácia maior na predição, pela versatilidade de uso e pelo maior suporte da comunidade, o Keras é uma opção melhor para o uso em ambientes corporativos e didáticos. Salvo as situações já citadas na seção 4.2.1.

Um possível trabalho futuro seria uma análise de viabilidade do Rubix, com outras linguagens populares no campo de aprendizado de máquina. Apesar de atualmente ele possuir uma limitação a respeito das redes neurais convolucionais, a implementação recente do NumPower, juntamente de esforços da comunidade, colocam no horizonte próximo a implementação de uma CNN com o Rubix. Por ser um projeto de código aberto, entusiastas da linguagem já estão contribuindo para essa funcionalidade.

Um estudo das possibilidades de uso do NumPower também pode abrir caminhos para o desenvolvimento de aprendizado de máquina com o PHP. Uma das maiores limitações da linguagem é a falta de suporte para a computação matemática avançada, e graças ao NumPower estas operações agora podem não só serem implementadas com facilidade mas também aceleradas pela GPU.

REFERÊNCIAS

ABADI, M. et al. {TensorFlow}: a system for {Large-Scale} machine learning. In: **12th USENIX symposium on operating systems design and implementation (OSDI 16)**. [S.l.: s.n.], 2016. p. 265–283. Citado 2 vezes nas páginas 16 e 23.

ACEMOGLU, D. et al. Artificial intelligence and jobs: Evidence from online vacancies. **Journal of Labor Economics**, The University of Chicago Press Chicago, IL, v. 40, n. S1, p. S293–S340, 2022. Citado na página 14.

ALMUSAED, A.; YITMEN, I.; ALMSSAD, A. Enhancing smart home design with ai models: A case study of living spaces implementation review. **Energies**, MDPI, v. 16, n. 6, p. 2636, 2023. Citado na página 14.

ANTHROPIC. **Anthropic 2024 Claude 3**. 2024. <https://www.anthropic.com/news/claude-3-family>. Accessed: 2024-07-18. Citado na página 14.

CHOLLET, F. et al. **Keras**. [S.l.]: GitHub, 2015. <https://github.com/fchollet/keras>. Accessed: 2024-07-18. Citado na página 25.

COGNITION. **Introducing Devin**. 2024. <https://www.cognition.ai/blog/introducing-devin>. Accessed: 2024-07-18. Citado na página 14.

FREDJ, H. B.; BOUGUEZZI, S.; SOUANI, C. Face recognition in unconstrained environment with cnn. **The Visual Computer**, Springer, v. 37, n. 2, p. 217–226, 2021. Citado na página 16.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado 2 vezes nas páginas 17 e 21.

GUIMARAES, M. G. O. **cnn-php**. 2024. Disponível em: <https://github.com/D4Cheap/cnn-php>. Citado na página 26.

GUIMARAES, M. G. O. **cnn-python**. 2024. Disponível em: <https://github.com/D4Cheap/cnn-python>. Citado na página 26.

HASHMI, F. **Face Recognition using Deep Learning CNN in Python**. 2021. Https://thinkingneuron.com/face-recognition-using-deep-learning-cnn-in-python/. Accessed: 2024-04-19. Citado na página 25.

HU, G. et al. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In: **Proceedings of the IEEE international conference on computer vision workshops**. [S.l.: s.n.], 2015. p. 142–150. Citado 2 vezes nas páginas 14 e 16.

ISHIKAWA, Y. **Rindow Neural Networks Documentation**. 2024. <https://rindow.github.io/neuralnetworks/index.html>. Accessed: 2024-07-18. Citado na página 14.

ISHIKAWA, Y. **Rindow Neural Networks Translation**. 2024. <https://rindow.github.io/neuralnetworks/index.html#note>. Accessed: 2024-07-18. Citado na página 31.

JIA, Y. et al. Caffe: Convolutional architecture for fast feature embedding. **arXiv preprint arXiv:1408.5093**, 2014. Citado na página 16.

KETKAR, N.; KETKAR, N. Introduction to keras. **Deep learning with python: a hands-on introduction**, Springer, p. 97–111, 2017. Citado na página 23.

KONDAS, A. **PHP-ML: Machine Learning library for PHP**. 2022. <https://php-ml.readthedocs.io/en/latest/>. Accessed: 2024-07-18. Citado na página 23.

LEARNING4J, D. **Deeplearning4j Suite Overview**. 2024. <https://https://deeplearning4j.konduit.ai/>. Accessed: 2024-04-19. Citado na página 16.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, p. 2278 – 2324, 12 1998. Citado 4 vezes nas páginas 14, 16, 19 e 22.

MATHWORKS. **Introduction to Deep Learning**. 2017. <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>. Accessed: 2024-04-19. Citado na página 22.

NUMPOWER. **Numpower: High-Performance Numeric Computing for PHP**. 2024. <https://numpower.org/>. Accessed: 2024-07-18. Citado na página 24.

POOLING. **Pooling**. 2019. <https://anhreynolds.com/blogs/cnn.html>. Accessed: 2024-04-19. Citado na página 21.

RAY, P. P. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. internet of things and cyber-physical systems, 3, 121–154. **URL https://doi. org/10.1016/j. iotcps**, v. 3, 2023. Citado na página 14.

RUBIX. **Rubix ML Documentation**. 2024. <https://docs.rubixml.com/latest/>. Accessed: 2024-07-18. Citado na página 14.

SCIENCE, T. D. **Computer vision**. 2024. <https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb6
Accessed: 2024-04-19. Citado na página 19.

TEAM, G. et al. Gemini: a family of highly capable multimodal models. **arXiv preprint arXiv:2312.11805**, 2023. Citado na página 14.

TECHS, W. **Usage statistics of server-side programming languages for websites**. 2024. <https://w3techs.com/technologies/overview/programming_language>. Accessed: 2024-04-19. Citado 2 vezes nas páginas 14 e 31.

VINAY, A. et al. G-cnn and f-cnn: Two cnn based architectures for face recognition. In: IEEE. **2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)**. [S.l.], 2017. p. 23–28. Citado na página 16.

VISIONX. **Visionx The Future of Machine Learning**. 2024. <https://visionx.io/blog/future-of-machine-learning-languages/>. Accessed: 2024-07-18. Citado na página 14.

XIE, Z.; LI, J.; SHI, H. A face recognition method based on cnn. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2019. v. 1395, n. 1, p. 012006. Citado 2 vezes nas páginas 16 e 22.

YANG, G. R. **Artificial neural networks for neuroscientists: A primer - Scientific Figure on ResearchGate.** 2016. <https://www.researchgate.net/figure/Comparing-the-visual-system-and-deep-convolutional-neural-networks-The-same-image-can-be_fig4_341817070>. Accessed: 2024-11-13. Citado na página 20.

ZHANG, A. et al. **Dive into Deep Learning**. [S.l.]: Cambridge University Press, 2023. <https://D2L.ai>. Citado 6 vezes nas páginas 14, 17, 18, 19, 20 e 21.