17º Congresso Interinstitucional de Iniciação Científica – CIIC 2023
29, 30 e 31 de agosto de 2023
ISSN: 2965-2812

DETECÇÃO DE PONTOS-CHAVE EM IMAGENS APLICADA À FENOTIPAGEM DE MILHO

Giovana Abe dos **Santos**[1]; Thiago Teixeira **Santos**[2]

Nº 23604

RESUMO *– Para o melhoramento genético de plantas, o acompanhamento de alterações na qualidade e saúde delas é importante, por isso, a fenotipagem de plantas, que exerce esse papel, é de extrema importância para a agricultura. Nesse sentido, o objetivo deste trabalho é auxiliar na fenotipagem de plantas de milho, um vegetal amplamente utilizado para diversas finalidades, desde a base de muitos alimentos humanos até para a criação de ração para animais. Tendo isso em vista, a utilização da visão computacional, por meio de redes neurais artificiais, é de grande ajuda, uma vez que ela pode realizar tarefas comumente feitas na fenotipagem de plantas de milho, como contagem de quantidade e estimação do tamanho das folhas das plantas, de forma rápida e eficiente em centenas de indivíduos. Desta forma, a criação de uma rede neural para detecção de pontos-chave (keypoints) nas folhas da planta, que permitirá gerar dados úteis para a contagem e estimativa de tamanho das folhas, está em desenvolvimento.*

Palavras-chaves: Visão computacional; aprendizado de máquina; detecção de keypoint; fenotipagem; redes neurais.

1 Autora, Bolsista CNPq (PIBIC): Graduação em Matemática Aplicada e Computacional, UNICAMP, Campinas-SP; giovana.abe@gmail.com.
2 Orientador: Pesquisador da Embrapa Agricultura Digital, Campinas-SP; thiago.santos@embrapa.br.

ISSN: 2965-2812

ABSTRACT – *For the genetic improvement of plants, tracking variations in their quality and health is important, therefore, plant phenotyping, which plays this role, is extremely important for agriculture. In this sense, the goal of this work is to assist the phenotyping of maize plants, a vegetable widely used for various purposes, from being the base of many human foods to the production of animal feed. With this in mind, the use of computer vision, through artificial neural networks, is helpful since it can perform tasks commonly performed in maize plants phenotyping, such as counting the quantity and estimating the size of maize plants leaves, quickly and efficiently in hundreds of individuals. To this end, the creation of a neural network for keypoint detection on plant leaves, in order to generate data that is useful for counting and estimating the size of the leaves, is under development.*

Keywords: Computer vision; machine learning; keypoint detection; phenotyping; neural networks.

1. INTRODUÇÃO

A fenotipagem de plantas é importante para o ramo de pesquisa em melhoramento genético, visto que se pode rastrear componentes da planta e observar se há alterações ou não quando ela está sujeita a determinada condição, genética ou de ambiente e, a partir disso, inferir maneiras de adquirir melhorias úteis (COSTA et al., 2019).

Antes do avanço tecnológico, a fenotipagem de plantas era, essencialmente, realizada de forma manual e, muitas vezes, destrutiva para a planta (COSTA et al., 2019). Além de ser menos prejudicial, a utilização da computação para detecção de elementos que auxiliem na fenotipagem de plantas é muito útil, pois, com ela é possível analisar uma grande quantidade de dados em um curto espaço de tempo, comparado à mesma tarefa realizada por humanos, e com uma boa precisão (DAS CHOUDHURY et al., 2019).

Tendo isso em vista, o estudo da fenotipagem de plantas com o auxílio da visão computacional, área da inteligência artificial que estuda maneiras do computador obter informações significativas a partir de imagens, vídeos e outras fontes visuais, está em crescimento. Essa intensificação pode ser observada com a quantidade de pesquisas na área, como por exemplo o

ISSN: 2965-2812

desenvolvimento contínuo de pacotes de software focados em fenotipagem como o PlantCV[3] (GEHAN e FAHLGREN, 2017), de código aberto e que já está em sua segunda versão, e o Phenotiki[4] (MINERVINI, 2017) que é também uma plataforma aberta de software e hardware para fenotipagem, baseada em plantas em forma de roseta.

Uma das avaliações contidas na fenotipagem de plantas é a contagem e estimação do tamanho das folhas de uma certa planta. Para o caso de plantas de milho, foco deste trabalho, essa identificação permite verificar o estágio de crescimento vegetativo da planta e utilizar esses dados para análises de melhoramento genético. A abordagem deste estudo para esse problema consiste em, por meio da fenotipagem por imagens e uso da visão computacional, tratá-lo como um problema de detecção de *keypoints*, pontos-chave, que permitam essa análise.

A detecção de *keypoints* pode ser feita por meio do uso de redes neurais artificiais, um modelo computacional de aprendizado de máquina, que aprende padrões específicos a partir de uma base de dados prévia e que, após o treinamento, podem fazer previsões em novos dados. Para o problema deste trabalho, foi feito o uso de redes neurais convolucionais, que são uma classe de redes neurais artificiais que funcionam bem com processamento de imagens.

A base de dados, *dataset*, utilizada para o treinamento de uma rede neural é uma parte extremamente importante, visto que é a partir dela que a rede poderá aprender os padrões requeridos e apresentar previsões precisas. Dessa forma, foi preciso construir uma base de dados, consistindo em imagens coloridas de plantas de milho em vários estágios de crescimento, com suas respectivas anotações dos *keypoints* nas bases e pontas de cada folha, e testado um modelo com base na rede *ResNet50*, criado na ferramenta Keras. Como os resultados obtidos com esse modelo, como mostrado neste trabalho, tiveram baixa acurácia, a principal contribuição deste trabalho até agora é a construção da base de dados com as anotações de *keypoints* e a escolha de possíveis caminhos para a experimentação de outros modelos.

2. MATERIAL E MÉTODOS

2.1. Construção da base de dados

A base de imagens utilizada neste projeto consiste em 79 imagens coloridas, cada uma

[3] PlantCV está disponível com sua documentação em: https://plantcv.danforthcenter.org/

[4] Phenotiki está disponível com sua documentação em: http://phenotiki.com/

ISSN: 2965-2812

contendo uma planta de milho em um vaso, sendo todas tiradas pela mesma câmera fotográfica, em diversos ângulos e iluminação distintos. O primeiro estágio de construção da base de dados, após ter a base de imagens, foi marcar os pontos-chave, *keypoints*, em todas as imagens.

Como o objetivo é identificar o número de folhas e estimar seus respectivos tamanhos, os *keypoints* anotados foram as bases e pontas (*bases* e *tips* em inglês) de todas as folhas da planta presentes nas imagens. Mais especificamente para o caso de plantas de milho, temos uma classificação para essas bases que é chamado de “colar” da folha (*collar* em inglês) que basicamente é uma área de cor mais clara localizada na base das folhas, como demonstra a Figura 1.

Figura 1. Imagem do colar de uma folha de planta de milho.

O colar não é formado para todas as folhas ao mesmo tempo, então o número de folhas com e sem colar é um indicativo do estágio de crescimento da planta.

Para fazer as anotações, foi utilizado o software *VGG Image Annotator (VIA)*, um programa de código aberto que permite fazer anotações em imagens manualmente. Para isso, foram criadas no software as classes *tip* e *collar* para, além de marcar os *keypoints*, já deixá-los classificados. Um exemplo de anotação feita pode ser vista na Figura 2.

Após fazer a anotação em todas as imagens da base, pode-se exportar um arquivo no formato JSON com todas as anotações feitas para utilizá-lo para treinar a rede neural posteriormente.

O arquivo JSON, com as anotações, foi construído no formato COCO, *Common Object in Context*. COCO é um dos *datasets* mais populares e nele estão disponíveis, publicamente, diversas imagens já rotuladas com variados objetos do dia-a-dia. Por conta disso, ele é amplamente utilizado para avaliar o desempenho de redes neurais na área de visão computacional.

ISSN: 2965-2812

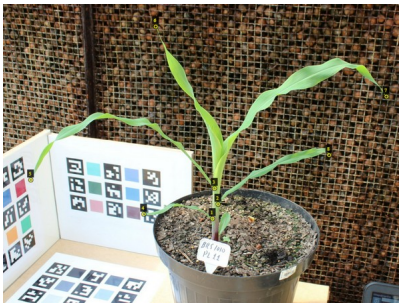

Figura 2. Exemplo de anotação feita no software *VIA*.

Por consequência, o formato COCO de anotações foi largamente adotado pelos pesquisadores do ramo e, pode-se observar a partir do exemplo abaixo, que esse formato é organizado em blocos e contém todas as informações necessárias do *dataset*, como as informações de cada imagem (nome, altura, largura e uma identificação), as anotações feitas com as coordenadas dos *keypoints*, o número de *keypoints* marcados, a identificação da imagem relacionada à anotação, coordenadas de uma caixa delimitadora (*bbox* - *bounding box*) que delimita a região onde existem *keypoints* anotados e, por fim, ainda há um bloco de categorias, que é útil para problemas que forem mais generalizados, com identificação de objetos distintos nas imagens, por exemplo.


```
{
    "images": [
        {
            "file_name": "IMG_0157.JPG",
            "height": 864,
            "width": 1296,
            "id": 1
        },
       ...
    ],
    "annotations": [
        {
            "segmentation": [
                []
            ],
            "keypoints": [
                771, 637, 2,
                     ...
            ],
            "num_keypoints": 7,
            "area": 439410.0,
            "iscrowd": 0,
            "image_id": 1,
            "bbox": [
                177.0,
                174.0,
                932.0,
                756.0
            ],
```

ISSN: 2965-2812


```
            "category_id": 1,
            "id": 1
        },
       ...
    ],
    "categories": [
        {
            "id": 1,
            "name": "maize"
        }
    ]
}
```


2.2. Tratamento dos dados e a rede neural utilizada

Para criar a rede neural artificial de aprendizado profundo, foi utilizada a biblioteca Keras em Python. Essa biblioteca tem como foco permitir a criação de redes de forma simples disponibilizando blocos de construção para isso, conseguindo transformar um projeto num resultado rapidamente.

No Keras, as redes neurais são implementadas no que é chamado de "modelo". Foi utilizado, para a construção do modelo, a "*Functional API*". Um modelo do Keras utilizando essa API consiste primordialmente em camadas de funções que tem como entrada um tensor e como saída um tensor com os dados de entrada transformados de alguma forma. Os tensores, na computação, são estruturas de dados que armazenam as informações que serão utilizadas no programa. Uma imagem colorida de tamanho 224×224, por exemplo, pode ser representada por um tensor de 3 dimensões da forma $(224, 224, 3)$, onde a primeira entrada representa a altura da imagem, a segunda a largura dela e a terceira a quantidade de canais de cores.

As operações com tensores realizadas pelas camadas de um modelo podem ser bem variadas, mas que, de forma geral, visam gerar uma nova representação do tensor de entrada. Cada nova representação serve para o modelo poder compreender e detectar padrões dos dados de entrada.

Para o presente trabalho, utilizou-se como *backbone* da rede a arquitetura *ResNet50*, onde foram carregados e congelados os pesos pré-treinados no *ImageNet* — banco de imagens amplamente utilizado para treinamento de modelos de visão computacional de larga escala —, que serão a base para o treinamento do modelo criado. O backbone de uma rede neural opera como se fosse uma sequência de filtros sobre o tensor de entrada, que tem como função fornecer, neste caso, uma representação da imagem que permita a discriminação dos padrões de interesse. Tendo isso em vista, a *ResNet* — *Residual Neural Network* —, que é uma rede neural convolucional muito

ISSN: 2965-2812

utilizada como *backbone* de modelos na área de visão computacional e bem conhecida por permitir lidar com o problema de gradientes decrescentes (*vanishing gradients*), comum em redes com muitas camadas (SOUZA et al., 2020), foi escolhida como *backbone* deste projeto.

Dessa forma, têm-se como primeira camada do modelo o *backbone*, que é o responsável por extrair as principais características dos dados (imagens) de entrada. Depois disso, foi incluída uma camada de *dropout*, uma técnica de regularização (CHOLLET, 2017) que aleatoriamente define como zero algumas características geradas pela camada anterior (tensor de saída dela). Em seguida, tem-se a camada de convolução, *SeparableConv2D*, que prepara o *input*, já processado, para gerar predições na última camada, de *output*. A quantidade de parâmetros do modelo e forma de *input* e *output* podem ser vistos na Tabela 1.

O *input* é um tensor que representa a imagem, uma imagem com 224 pixels de altura e largura e com 3 canais de cores (R - *red*/vermelho, G - *green*/verde e B - *blue*/azul). O *output* é um tensor 1 por 1 com 48 entradas, que representam os 24 *keypoints* de cada imagem.

Vale destacar que cada imagem contém 24 dados de *keypoints* anotados, pois as imagens presentes na base de dados deste projeto contém plantas de milho até o estágio de crescimento vegetativo V8 — plantas que contém 8 folhas com colar formado —, mas como há a possibilidade de existirem folhas sem o colar ainda, considerou-se um máximo possível de 12 folhas com ou sem colar. Os pontos presentes numa determinada imagem foram anotados com sua respectiva posição em coordenadas x e y com um indicativo de visibilidade — 1, se visível — $(x, y, 1)$ e, para completar os 24 pontos, foram adicionados entradas $(0, 0, 0)$. Isso porque a rede neural treinada deve ter um tamanho de *output* fixo.

É importante salientar que, antes do treinamento, a base de dados foi dividida em 3 grupos: conjunto de treinamento (50 imagens), de validação (10 imagens) e de teste (19 imagens). Isso é feito para não haver vazamento de dados entre o treinamento, a validação do modelo e o teste real feito no final do processo de ajuste no modelo. O vazamento pode ocorrer, por exemplo, caso tenha o mesmo dado em mais de um dos conjuntos, assim, para que o conjunto de teste permita avaliar a generalidade do modelo para novas plantas, esses conjuntos devem ser disjuntos. Um vazamento de informações pode gerar resultados artificialmente positivos no conjunto de testes, superestimando a capacidade de generalização do modelo quando estiver em operação. Como a base de dados disponível para trabalhar é pequena, a utilização de aumentações fez-se necessária, visto que redes de aprendizado profundo precisam de muitos dados para obter bons resultados e prevenir um *overfitting* — sobreajuste, o que significa que o modelo aprendeu apenas

ISSN: 2965-2812

os padrões específicos dos dados de treinamento e é ineficiente quando usado para prever novos resultados.

Tabela 1. Tipo, forma e quantidade de parâmetros de cada camada da rede neural.

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 224, 224, 3)]	0
tf.__operators__.getitem (SlicingOpLambda)	(None, 224, 224, 3)	0
tf.nn.bias_add (TFOpLambda)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
dropout (Dropout)	(None, 7, 7, 2048)	0
separable_conv2d (SeparableConv2D)	(None, 3, 3, 48)	149552
separable_conv2d_1 (SeparableConv2D)	(None, 1, 1, 48)	2784
Total params: 23,740,048 **Trainable params: 152,336** **Non-trainable params: 23,587,712**		

O processo de aumentação da base de dados consiste em criar novas amostras de dados para o treinamento utilizando as já existentes. No caso de imagens, esse procedimento pode facilmente ser visto como um espelhamento de uma imagem do conjunto de treinamento, uma rotação, alteração do brilho, recorte de apenas parte da imagem, leve alteração de cores, entre outras transformações possíveis que devem ser escolhidas de acordo com o problema.

Esse mecanismo foi feito, neste trabalho, com o auxílio da biblioteca Albumentations[5] no Python. Para o caso estudado de imagens de plantas de milho, as aumentações utilizadas foram: recorte aleatório, alteração do brilho e contraste aleatório, espelhamento horizontal, alteração da perspectiva da câmera, rotação da imagem e, por último, um redimensionamento para deixar todas as imagens do mesmo tamanho para serem utilizadas como *input* do modelo. Vale ressaltar que, como o problema é de detecção de *keypoints*, é preciso estar atento às aumentações da biblioteca que funcionam para *keypoints*, ou seja, que alterem as posições deles de forma equivalente à transformação feita nas imagens.

[5] Mais informações sobre a biblioteca Albumentations podem ser encontradas no site: https://albumentations.ai/

ISSN: 2965-2812

3. RESULTADOS E DISCUSSÃO

O modelo criado foi compilado utilizando o otimizador Adam (Estimativa de Momento Adaptativo) — algoritmo de otimização baseado no método do gradiente descendente estocástico (SGD - stochastic gradient descent) — e como função perda, *loss function*, a MAE (Mean Absolute Error - Erro Médio Absoluto). A função perda essencialmente indica por meio de um único número o quão distante a previsão da rede está com a resposta esperada. O otimizador tem como propósito minimizar a função perda, ou seja, ele guia a rede a cada passo do treinamento para ela convergir para um resultado ótimo, atualizando os parâmetros utilizados nas funções aplicadas aos tensores em cada camada do modelo.

Após as anotações, tratamento dos dados e compilação do modelo, foi possível treinar a rede neural criada. O treinamento foi realizado com 100 épocas e, como há poucas imagens disponíveis na base de dados utilizada, o tamanho escolhido para cada lote (*batch*) foi 4. O número de épocas é basicamente a quantidade de vezes que todo o conjunto de dados será visitado pela rede, em busca do menor valor para a *loss function*, e um *batch* é uma porção do conjunto de dados que passará na rede a cada iteração, por limitações de memória que impedem processar tudo de uma só vez. Em outros termos, o conjunto de treinamento é dividido em *batches* de um tamanho específico e eles passarão pela rede de treinamento atualizando os parâmetros, cada *batch* que passa pela rede é chamado de uma iteração e, uma época é quando todos os dados do conjunto (todos os *batches*) passaram pela rede.

Logo em seguida ao treinamento da rede, foi gerado um gráfico com os valores da função perda com o passar das épocas, já que é ela um dos indicativos normalmente utilizado para analisar a qualidade do ajuste do modelo aos dados. O gráfico pode ser visto na Figura 3.

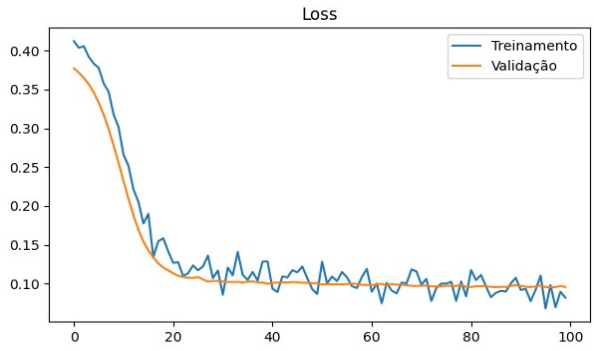

Figura 3. Gráfico com a relação da função perda ao passar das épocas de treinamento.

ISSN: 2965-2812

O resultado mostrado no gráfico é um bom indicativo de que a rede se ajustou bem aos dados e foi capaz de gerar boas previsões nos dados de validação também, visto que o valor da *loss function* da validação seguiu o decaimento do treinamento. Esse gráfico tem o comportamento esperado para bons resultados de redes neurais, onde a função perda no treinamento decai e a validação também tem um decaimento semelhante.Assim, era esperado a obtenção de boas previsões para as detecções dos *keypoints*, contudo, ao selecionar algumas amostras do conjunto de validação, pode-se perceber que as previsões não foram tão boas. Dois exemplos podem ser vistos nas Figuras 4 e 5, sendo, da esquerda para direita, a imagem base, a imagem com os *keypoints* anotados (resultado esperado) e a imagem com as previsões do modelo:

Figura 4. Exemplo 1 - imagem da planta com seus respectivos resultados esperados e previsões.

Ao revisar as previsões geradas, vê-se que há a possibilidade de que os *keypoints* previstos próximos a $(x, y)=(0,0)$, que coincidem com o esperado, podem ter contribuído bastante para a função perda fornecer um valor baixo, indicando, assim, que o resultado da rede é bom apesar dos outros pontos previstos não serem realmente próximos do aguardado.

Figura 5. Exemplo 2 - imagem da planta com seus respectivos resultados esperados e previsões.

ISSN: 2965-2812

4. CONCLUSÃO

O presente projeto é o início da criação de uma ferramenta que será útil e vantajosa para o setor de melhoramento genético de plantas de milho. Espera-se que após a ferramenta estar apta a classificar as fases de desenvolvimento de milho, ela possa ser ajustada para auxiliar outros setores de melhoramento de vegetais diferentes.

Como o resultado da rede neural criada e treinada com a ferramenta Keras não produziu os resultados com a acurácia esperada, algumas hipóteses foram levantadas e concluiu-se que outra ferramenta deve ser utilizada para a continuidade do projeto. Isto, mesmo utilizando uma boa arquitetura de rede disponível na Keras, boas imagens e anotações, além de muitas aumentações com o auxílio da biblioteca *Albumentations*, o objetivo não foi alcançado de forma precisa.

Para o progresso do trabalho subsequente, será experimentada a ferramenta MMPose, que é uma "*toolkit*" (caixa de ferramentas em português) para estimativas de poses ─ humanas, de animais ─ baseada em PyTorch, que usa como base a biblioteca de código aberto de aprendizado de máquina Torch, em Python. Para a utilização dos algoritmos disponíveis na MMPose, é preciso que as anotações dos *keypoints* estejam num formato específico, COCO, e essas já começaram a ser desenvolvidas durante o período deste trabalho.

Vale destacar que a MMPose dispõe implementações de modelos no estado-da-arte para detecção de keypoints, isto é, modelos que acompanham o estado atual de desenvolvimento nessa área. Um exemplo disso pode ser visto no artigo "*Simple Baselines for Human Pose Estimation and Tracking*" (XIAO et al., 2018), que utiliza implementações que estão disponíveis na MMPose e uma arquitetura de organização semelhante também. Essa pesquisa reúne uma base simples e funcional, a partir de diversos outros trabalhos no estado-da-arte, de detecção de *keypoints* para a estimativa de posições de humanos e rastreio delas quadro a quadro para o caso de vídeos. Esse artigo demonstra, ainda, que se pode obter um bom desempenho com um modelo simples, visto que o modelo apresentado tem linhas de base simples, mas fortes, que geraram resultados de ponta e até melhores que outros modelos no estado-da-arte vencedores de desafios como "*COCO 2017 Keypoint Detection Task*" e "*PoseTrack Challenge*" na Conferência Internacional sobre Visão Computacional de 2017 (ICCV 2017).

Além disso, foi levantada a hipótese de construir uma base de dados maior, visto que o treinamento de redes neurais convolucionais de visão computacional normalmente requerem um

ISSN: 2965-2812

grande número de dados de treinamento para produzirem melhores resultados. Ademais, também está sendo analisada a possibilidade de uso de outras funções perda.

5. AGRADECIMENTOS

Agradecimento ao Programa Institucional de Bolsas de Iniciação Científica, PIBIC, do Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, pela concessão da bolsa.

6. REFERÊNCIAS

CHOLLET, F. **Deep learning with Python**. 2017. New York, NY: Manning Publications.

COSTA, C; SCHURR, U; LORETO, F; MENESATTI, P; CARPENTIER, S. **Plant Phenotyping Research Trends, a Science Mapping Approach**. 2019. Artigo - Frontiers in Plant Science.

DAS CHOUDHURY, S; SAMAL, A; AWADA, T. **Leveraging Image Analysis for High-Throughput Plant Phenotyping**. 2019. Artigo - Frontiers in Plant Science.

GEHAN, M. A; FAHLGREN, N. et al. **PlantCV v2: Image analysis software for high-throughput plant phenotyping**. 2017. Artigo - PeerJ.

MINERVINI, M. **Phenotiki: an open software and hardware platform for affordable and easy image-based phenotyping of rosette-shaped plants**. 2017. Artigo - The Plant journal: for cell and molecular biology.

SOUZA, V; SILVA, L; ARAUJO, L; SANTOS, A. **Análise Comparativa de Redes Neurais Convolucionais no Reconhecimento de Cenas**. Disponível em: <http://dx.doi.org/10.14210/cotb.v11n1.p419-426>. Acesso em: 12 jun. 2023. 2020. Anais do XI Computer on the Beach.

XIAO, B.; WU, H.; WEI, Y. **Simple baselines for human pose estimation and tracking**. Disponível em: <http://arxiv.org/abs/1804.06208>. Acesso em: 03 jul. 2023. 2018. European Conference on Computer Vision.