Learning CNN Filters From User-Drawn Image Markers for Coconut-Tree Image Classification

Italos Estilon de Souza and Alexandre Xavier Falcão, *Member, IEEE*

Abstract—Identifying species of trees in aerial images is essential for land-use classification, plantation monitoring, and impact assessment of natural disasters. The manual identification of trees in aerial images is tedious, costly, and error-prone, so automatic classification methods are necessary. Convolutional neural network (CNN) models have well succeeded in image classification applications from different domains. However, CNN models usually require intensive manual annotation to create large training sets. One may conceptually divide a CNN into convolutional layers for feature extraction and fully connected layers for feature space reduction and classification. We present a method that needs a minimal set of user-selected images to train the CNN's feature extractor, reducing the number of required images to train the fully connected layers. The method learns the filters of each convolutional layer from user-drawn markers in image regions that discriminate classes, allowing better user control and understanding of the training process. It does not rely on optimization based on backpropagation, and we demonstrate its advantages on the binary classification of coconut-tree aerial images against one of the most popular CNN models.

Index Terms—Design of convolutional neural networks (CNNs), interactive machine learning, remote sensing image analysis.

I. INTRODUCTION

DEEP learning has proven to be applicable to different tasks, from image classification to data synthesis [1]. In remote sensing, the applications may involve segmentation of terrain images [2], [3], building identification [4]–[6], and deforestation monitoring [7], for instance. In this work, we are interested in identifying species of trees from aerial images. The topic is important for land-use classification, plantation monitoring, and damage assessment of natural disasters. As the plantations can span vast areas, the manual identification of each tree is costly, tedious, and error-prone, so automatic classification methods are necessary.

Classification of tree species in aerial images has been actively investigated [8]. Puttemans *et al.* [9] and Hebbar *et al.* [10] present automatic solutions to detect coconut trees based on convolutional neural network (CNN) models. Despite these recent advances, CNN models usually require considerable human effort in image annotation to create large training sets. Vargas-Muñoz *et al.* [11] proposed an active learning approach to mitigate the problem. The method explores data projection techniques to allow simultaneous annotation of multiple regions as having or not coconut trees. It then uses CNN to identify the candidate regions with coconut trees. We adopt another alternative—the design of simplified CNN models from small training sets with user interaction, being the user knowledgeable about CNN modeling and the application domain.

The design of CNN models without a human as a part of the training loop leaves several questions unanswered: 1) how to find a useful and simplified model for a given classification problem? 2) how to train that model from a minimum number of annotated images? 3) can the user explain the decisions of the model? and 4) can the model improve from label corrections? The first question requires human knowledge about CNN modeling and the problem of interest. The second one requires to reduce the human effort to train a CNN model. The third issue is related to human understanding. It may explore visual analytics to explain the model's decisions and guide the user in the design of the model [12]–[14]. The fourth question is also essential during training, and it is related to user control over the process. They all lead to the importance of involving human experts during the machine learning process. Explainable artificial intelligence (XAI) has appeared to address (3) [15], [16]. However, the importance of human participation during the training process has called little attention [17]. Exceptions rely on visual analytics [12]–[14], and we present here a first work in which the user has direct influence over the quality of the filters.

First, we conceptually divide a CNN into convolutional layers for feature extraction and fully connected layers for feature space reduction and classification. Each convolutional layer contains a filter bank, an activation function, and alternative operations (e.g., pooling and batch normalization). As the number of convolutional layers increases (deeper is the model), higher is the number of annotated images required to train the model by backpropagation.

To reduce the need for large annotated training sets by exploiting the user knowledge, we present a method, called feature learning from image markers (FLIM), that needs a minimal set of images to learn the filters of each convolutional layer. The user selects the number of convolutional layers, their filter sizes, and a few images from a training set and draws markers in image regions that best discriminate the classes. The user takes a couple of minutes to select images and draw

markers in those images. By clustering patches extracted from marker pixels at the input of each convolutional layer, FLIM finds filters that enhance those regions at the convolutional layers' output. The remaining training images can then be used to train the fully connected layers or another classifier. The CNN architecture could also be optimized with the support of visual analytics [14], but we have not explored this option yet. In interactive semantic segmentation, user-selected clicks can affect a part of the network to improve segmentation, but the filters are not designed from those markers [18].

We demonstrate the advantages of this interactive technique on the binary classification of coconut-tree images against VGG-16 [19]. First, the user can better understand and control the training process by observing the effectiveness of marker selection. Second, very few images per class (e.g., less than five) seem to be enough. Third, by eliminating the optimization of the convolutional layers by backpropagation, FLIM can reduce the number of annotated images to train the fully connected layers. Fourth, FLIM is application-independent, so it might be useful for other image classification problems.

II. FLIM: Feature Learning From Image Markers

A CNN model should enhance image regions that best discriminate classes through a sequence of convolutional layers. This section shows how identifying such regions by a simple user action can make that sequence an effective feature extractor. The method is named FLIM. In the feature space that results from convolutional layers, it is common to assume that the classes are piecewise linearly separable and apply fully connected layers—multilayer perceptron (MLP)—for feature space reduction and classification. The training of a CNN usually relies on weight optimization using backpropagation, which requires a higher number of annotated images as deep as the model is. In FLIM, we eliminate the need for backpropagation to train convolutional layers by finding a set $\mathcal{F}$ of filters that can enhance discriminant regions from all classes.

The user selects a few images from a training set to compose a very small data set $\mathcal{D}$ and draws labeled markers in image regions that best discriminate the classes. The convolution between an image I and a filter $F \in \mathcal{F}$ enhances the regions whose local patterns are detected by F. We wish then to estimate the weights of F such that those local patterns are characteristic of one given class. For the sake of clarity, one must interpret the convolution operation at a pixel p of I as the inner product $\text{vec}(P_I(p)) \cdot \text{vec}(F)$, where $P_I(p)$ is a $k \times k$ patch with m bands around p, F is a filter with shape $k \times k \times m$, and vec is the vectorization operation. A filter F can discriminate a class $i \in \{1, 2, \ldots, c\}$ among c classes when it generates positive results for local patterns $\text{vec}(P_I(p))$ of class i and negative results for patterns from other classes. That is, $\text{vec}(F)$ is the normal vector of a hyperplane in $\mathbb{R}^{k \times k \times m}$, which detects the patterns of class i at its positive side. We wish to estimate $\mathcal{F}$ such that its filters will detect patterns from all classes in different positions p.

For a problem with c classes where $\lambda(p) = i \in \{1, 2, \ldots, c\}$ is the label of a marker pixel p from class i, let $\mathcal{M}_I$ be a set of marker pixels drawn in image $I \in \mathcal{D}$ and $P_I(p)$ be the respective patch around a pixel $p \in \mathcal{M}_I$. Let $\mathcal{P}_i$ be the set of all patches around marker pixels from all
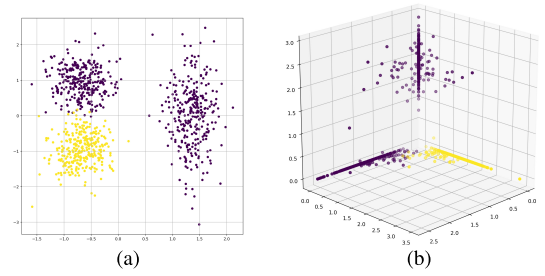

Fig. 1. (a) 2-D feature space with color-coded samples from two classes distributed in three groups. (b) After marker-based batch normalization and convolution with filters obtained from each group, the classes can be more easily detected by two filters in a new 3-D feature space.

images $I \in \mathcal{D}$, with representative examples $\text{vec}(P_I(p))$ from class $i \in \{1, 2, \ldots, c\}$

$$\mathcal{P}_i = \bigcup_{I \in \mathcal{D}, p \in \mathcal{M}_I, \lambda(p) = i} P_I(p). \tag{1}$$

A clustering operation on each set $\mathcal{P}_i$, $i \in [1, c]$, guarantees groups with similar local patterns for each class. The groups must be shifted to the origin of $\mathbb{R}^{k \times k \times m}$ by subtracting the mean value of all patches and set $\mathcal{P} = \bigcup_{i \in \{1,2,\ldots,c\}} \mathcal{P}_i$. In addition, the standard deviation of all patches is computed and used for standardization. This operation allows batch normalization of image sets using the mean and standard deviation of the patches from the image markers, i.e., a marker-based batch normalization. The centroid of each cluster defines the weights of each filter $F \in \mathcal{F}$, and we force norm $\| \text{vec}(F) \| = 1$ to avoid preferences among them. The centralization for filter definition is paramount to eliminate activations in regions from other classes, whose patterns fall in the negative side of the hyperplane with normal $\text{vec}(F)$. Fig. 1 illustrates three groups with color-coded samples from two classes in a hypothetical 2-D feature space. The marker-based batch normalization and convolution with filters obtained as centroids of the three groups in Fig. 1(a) create a new 3-D feature space [see Fig. 1(b)], in which the classes can be more easily detected by two filters in distinct activation bands.

Each convolutional layer is trained individually, one layer at a time, to find its filter set $\mathcal{F}$. The number of filters per layer depends on the clustering technique. We use K-means, and then, the user must specify the number $K = |\mathcal{F}|$ of filters. After the convolution operation, we apply the ReLU function to eliminate negative activations and the max-pooling operation to aggregate local information. Note that, apart from the initial marker selection, the training process is automatic. We have preserved the convolutional layers' output dimensions and the marker pixels, so we can find the filters of layer $L + 1$ from the output of layer L. Fig. 2 shows one example of a simple network projected by FLIM for the experiments of Section III.

III. Experiments

We use a data set from [11], which contains regions of aerial images with and without coconut trees from the Kingdom of Tonga, as acquired in October 2017.[1] Each region is classified as containing one or none coconut trees, but part

[1] The images are available in blog.werobotics.org/2018/01/11/open-ai-challenge-2.
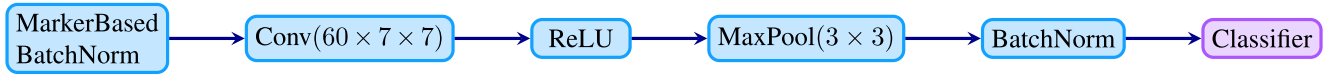
Fig. 2. Example of a network with a single convolutional layer followed by a classifier, as projected by FLIM, and used for the experiments in this work.
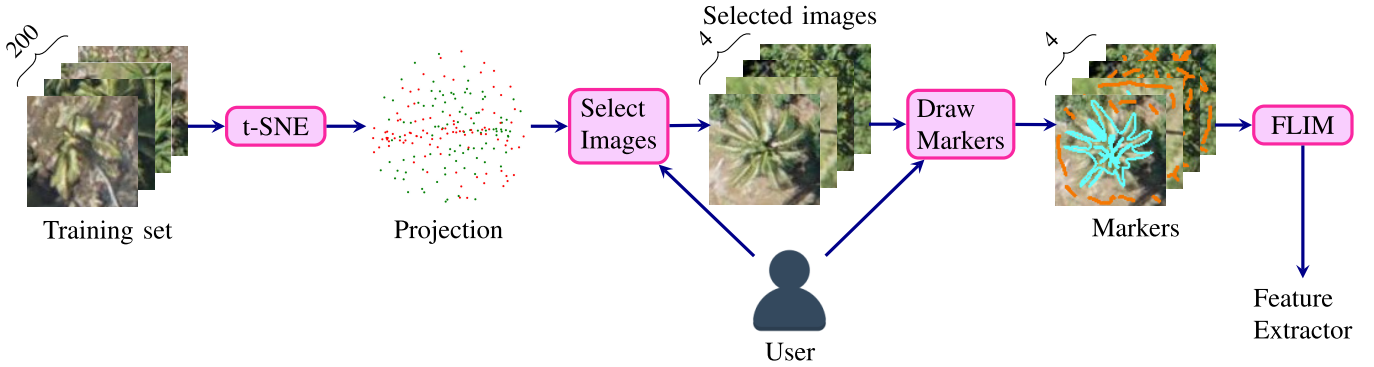
Fig. 3. Pipeline for training a feature extractor using FLIM. In the projection, green points are images with coconut trees, and red points are images with no coconut trees. The user selects a few images (e.g., 4) from the projection. The user draws orange markers on background pixels and cyan markers on coconut-tree pixels for image feature learning by FLIM.

of coconut trees from adjacent regions might appear near the region's border. The Humanitarian OpenStreetMap community annotated the regions. The data set consists of 13 587 regions with 90 × 90 pixels and spatial resolution of 8 cm, being 10 268 and 3319 regions annotated as containing and not containing coconut trees, respectively. The region images contain three bands: red, green, and blue. We convert them to the CIE L*a*b* color space and normalize each band within [0, 1].

The experiments used an Intel Xeon CPU E5-2620 v4 @ 2.10-GHz CPU and a Titan X with 12 GB of memory. They have been designed to evaluate the following hypotheses about FLIM.

H1: It can produce competitive feature extractors from user-drawn markers in a few images selected per class from a training set.

H2: For a given network architecture, FLIM can provide better classification results than those obtained by training the same architecture with backpropagation.

H3: Its feature extractor remains competitive independently of the training set size.

User-drawn

H4: Markers on a higher number of selected images can improve feature extraction.

For these experiments, we have selected the following models that combine feature extraction and classification methods.

1) *FLIM+SVM:* Image classification is performed by support vector machines (SVMs) from FLIM-based features.

2) *FLIM+MLP:* Image classification is performed by MLP from FLIM-based features.

3) *FLIM-FT+MLP:* The architecture in FLIM+MLP is fine-tuned by backpropagation.

4) *CL+MLP:* The architecture in FLIM+MLP is trained from scratch by backpropagation.

5) *VGG:* VGG-16 [19] is trained from scratch.

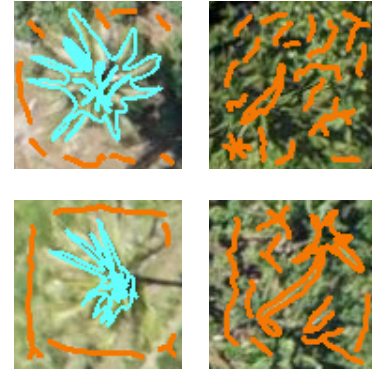6) *VGG-FT:* VGG-16 is pretrained on ImageNet and fine-tuned on the coconut-tree training set.

Fig. 4. Markers used for training, cyan markers on coconut-tree pixels, and orange markers on background pixels.

7) *VGG-FE+SVM:* Image features are obtained from the last convolutional layer of VGG-FT.

First, we evaluated H1 and H2 by randomly selecting 200 images for the training set and 2000 images for a validation set, and the remaining 11 387 images were left in the testing set. The validation set was used only once to choose empirically the architecture of the network shown in Fig. 2 and to choose the training hyperparameters. The convolutional layer has 60 filters with dimension 7 × 7 and three color bands. It includes marker-based normalization before convolution, ReLu activation, and a max-pooling operation with a window of dimension 3 × 3. Since it is the single and last convolutional layer, we applied a stride of 4 in max-pooling and batch normalization to create an input as close as possible to the one of the MLP classifier used in the popular VGG-16 [19]. We then vectorized and projected the training images by t-SNE [20] (see Fig. 3). Four representative images, only two per class, were chosen by the user to draw markers (see Fig. 4). These images must come from regions in the projection populated by samples from the same class (either red or green points). Those regions indicate groups of images from the same class in the input feature space. The markers must be placed in regions that best distinguish the coconut and noncoconut classes.

TABLE I

MEAN AND STANDARD DEVIATION OF PRECISION, RECALL, AND F-SCORE
FOR EACH METHOD: A FEATURE EXTRACTOR WITH A CLASSIFIER

Method	Precision	Recall	F-score
FLIM+SVM	0.856 ±0.011	0.831 ±0.019	0.838 ±0.017
VGG-FE+SVM	0.855 ±0.001	0.816 ±0.007	0.826 ±0.006
FLIM+MLP	**0.863 ±0.002**	**0.849 ±0.005**	**0.854 ±0.004**
FLIM-FT+MLP	0.845 ±0.003	0.825 ±0.006	0.832 ±0.005
CL+MLP	0.822 ±0.011	0.805 ±0.013	0.811 ±0.012
VGG	0.827 ±0.003	0.770 ±0.016	0.784 ±0.014
VGG-FT	**0.872 ±0.007**	**0.844 ±0.015**	**0.851 ±0.014**

TABLE II

MEAN AND STANDARD DEVIATION OF PRECISION, RECALL, AND F-SCORE
FOR EACH METHOD: A FEATURE EXTRACTOR TRAINED WITH EIGHT
IMAGES AND AN SVM CLASSIFIER

Method	Precision	Recall	F-score
FLIM+SVM (4)	0.856 ±0.011	0.831 ±0.019	0.838 ±0.017
FLIM+SVM (8)	0.857 ±0.005	0.838 ±0.010	0.844 ±0.009
FLIM+SVM (12)	0.854 ±0.005	0.839 ±0.009	0.844 ±0.008

Pytorch was chosen to implement FLIM and MLP, being SVM the linear model available in scikit-learn with $C = 0.01$ [21], [22]. We trained MLP for 40 epochs and the whole network for 60 epochs with a batch size of 64. We used a learning rate of 0.0001 and a weight decay of 0.001, and we updated the learning rate by a factor of 0.1 at every 5 epochs after epoch 30. VGG and VGG-FT were trained for 95 epochs with a batch size of 64, a learning rate of 0.001, and a weight decay of 0.001, and we updated the learning rate by a factor of 0.1 at every 5 epochs after the model achieved accuracy 0.95.

Table I shows the mean results of this first experiment for three random splits of the data set into the training, validation, and testing sets. We considered, better (in bold), the results with nonoverlapping intervals of standard deviation around the mean value. The comparison between FLIM+SVM and VGG-FE+SVM indicates that FLIM can produce a feature extractor with a single-layer competitive to VGG-FE using 13 layers (i.e., FLIM uses only 0.07% of the number of parameters in VGG-FE). The results of FLIM+MLP, CL+MLP, and FLIM-FT+MLP show that FLIM may dismiss training and fine-tuning based on backpropagation, by directly providing a more effective solution than CL+MLP and FLIM-FT+MLP (H2). This result can also be noticed when comparing FLIM+MLP with VGG, confirming that FLIM can reduce the number of required training images for the MLP classifier. The comparison between FLIM+MLP and VGG-FT shows competitive results with a top-rated solution (H1).

To evaluate H3, we used the validation set to increase the classifiers' number of training images. Note that the marker selection for FLIM was still the same from those four images. To our surprise, as the training set increases, the differences in mean f-score, as obtained among the three splits, favor FLIM+MLP over FLIM-FT+MLP (H2), VGG-FT (H1 and H3), and CL+MLP (H2), as shown in Fig. 7. The differences between FLIM+MLP and CL+MLP demonstrate that FLIM is superior to backpropagation in filter estimation (H2). Finally, Table II shows the mean results of FLIM+SVM among the three splits as the number of images for marker selection increases, and the training set size remains with

Fig. 5. Misclassified images by "FLIM + MLP" (first row) and VGG-FT (second row), being the last column with images without coconut trees.
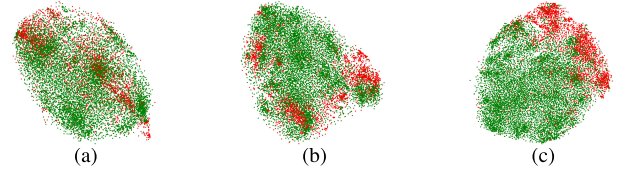

(a)　　　　　　　(b)　　　　　　　(c)

Fig. 6. Projections by t-SNE of the test images at (a) input layer, (b) output of the convolutional layer of FLIM + MLP after batch normalization (see Fig. 2), and (c) output of the last hidden layer of the MLP classifier in FLIM+MLP.
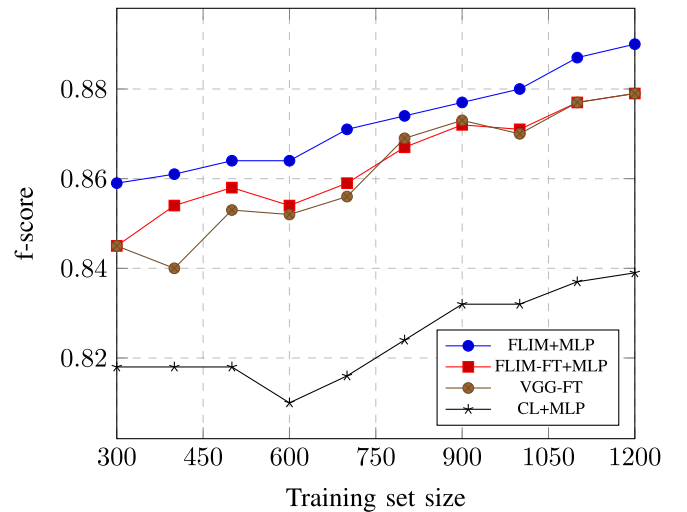

Fig. 7. Graph showing the impact of training set size on quantitative results. FLIM + MLP feature extractor was trained only with four images, and the remaining training set was used to train the MLP classifier.

200 images (H4). The results are not conclusive for H4, but the choice of more markers does not deteriorate FLIM's performance. This point deserves further investigation, i.e., choosing relevant training images and markers for FLIM.

In Fig. 5, we can see examples of misclassified images. In these images, coconut trees appear in different angles, sizes, and shapes, or the boundaries between the coconut tree and the background are tenuous, making it more difficult to identify them. In turn, images that do not contain coconut trees resemble their shape. The network designer could add markers in those images to improve the feature extractor in an active learning loop.

As motivated by Rauber *et al.* [15], we created projections of three stages of the network FLIM+MLP using t-SNE [20] to understand how our CNN transforms image spaces along with its layers. In Fig. 6(a), we can see the projection of the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

DE SOUZA AND FALCÃO: LEARNING CNN FILTERS FROM USER-DRAWN IMAGE MARKERS 5

test images in the CIE L*a*b* color space. Green points are images that contain coconut trees, and red points are images that do not contain coconut trees. The projection shows considerable overlapping between the two classes. The samples of both classes are also dispersed in the projection. In Fig. 6(b), the projection of the output of our feature extractor shows some reduction in sample dispersion. The classes show more concentration in some parts of the projection and reduced overlapping between them. Finally, Fig. 6(c) shows the output of the last hidden layer of the MLP classifier in FLIM+MLP. The overlapping between classes and sample dispersion is considerably reduced, explaining our results.

IV. CONCLUSION

We introduced a first feature learning technique, named FLIM, that can estimate effective filter weights for the convolutional layers of given network architecture from user-drawn markers in a few images of a training set. The resulting feature extractor can be used with different classifiers, and when it is used with an MLP classifier, the number of required images to train the fully connected layers is reduced. We demonstrated the advantages of FLIM over solutions based on VGG-16 for the binary classification of coconut-tree images. The experiments indicated that FLIM is an effective approach to produce considerably simplified network architectures. By involving the user in the training process, FLIM improves the understanding of CNN models and user control over the training process.

We intend to further investigate FLIM for different applications, image and marker selection for FLIM, elaborate methodologies to optimize network architecture, and extend FLIM to estimate weights in fully connected layers.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] R. Kemker, C. Salvaggio, and C. Kanan, "Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning," *ISPRS J. Photogramm. Remote Sens.*, vol. 145, pp. 60–77, Nov. 2018.

[3] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen, "Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2016, pp. 1–9.

[4] Y. Xu, L. Wu, Z. Xie, and Z. Chen, "Building extraction in very high resolution remote sensing imagery using deep learning and guided filters," *Remote Sens.*, vol. 10, no. 1, p. 144, Jan. 2018.

[5] T. Lu, D. Ming, X. Lin, Z. Hong, X. Bai, and J. Fang, "Detecting building edges from high spatial resolution remote sensing imagery using richer convolution features network," *Remote Sens.*, vol. 10, no. 9, p. 1496, Sep. 2018.

[6] Y. Liu *et al.*, "Multilevel building detection framework in remote sensing images based on convolutional neural networks," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 10, pp. 3688–3700, Oct. 2018.

[7] L. Bragilevsky and I. V. Bajic, "Deep learning for amazon satellite image analysis," in *Proc. IEEE Pacific Rim Conf. Commun., Comput. Signal Process. (PACRIM)*, Aug. 2017, pp. 1–5.

[8] F. E. Fassnacht *et al.*, "Review of studies on tree species classification from remotely sensed data," *Remote Sens. Environ.*, vol. 186, pp. 64–87, Dec. 2016.

[9] S. Puttemans, K. Van Beeck, and T. Goedemé, "Comparing boosted cascades to deep learning architectures for fast and robust coconut tree detection in aerial images," in *Proc. 13th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2018, pp. 230–241.

[10] R. Hebbar, P. Aparna, M. P. Harshita, H. Sounder, K. Nandkishore, and P. V. Vinod, "CNN based technique for automatic tree counting using very high resolution data," in *Proc. Int. Conf. Design Innov. 3Cs Compute Communicate Control (ICDI3C)*, Apr. 2018, pp. 127–129.

[11] J. E. Vargas-Munoz, P. Zhou, A. X. Falcao, and D. Tuia, "Interactive coconut tree annotation using feature space projections," in *Proc. IGARSS-IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2019, pp. 5718–5721.

[12] P. E. Rauber, A. X. Falcão, and A. C. Telea, "Projections as visual aids for classification system design," *Inf. Visualizat.*, vol. 17, no. 4, pp. 282–305, Oct. 2018.

[13] R. Garcia, A. X. Falcao, A. C. Telea, B. C. da Silva, J. Torresen, and J. L. D. Comba, "A methodology for neural network architectural tuning using activation occurrence maps," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–10.

[14] T. Spinner, U. Schlegel, H. Schäfer, and M. El-Assady, "Explainer: A visual analytics framework for interactive and explainable machine learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 1, pp. 1064–1074, Jan. 2020.

[15] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea, "Visualizing the hidden activity of artificial neural networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 101–110, Jan. 2017.

[16] A. B. Arrieta *et al.*, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[17] P. Angelov and E. Soares, "Towards explainable deep neural networks (xDNN)," *Neural Netw.*, vol. 130, pp. 185–194, Oct. 2020.

[18] K. Sofiiuk, I. Petrov, O. Barinova, and A. Konushin, "F-BRS: Rethinking backpropagating refinement for interactive segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8623–8632.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[20] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[21] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[22] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.