Fine-Grailed Classification Applied to Foliar Diseases in Apple Trees

Luca Rettenberger
Ravensburg-Weingarten
University of Applied Sciences
lr-192176@rwu.de

Abstract

In this work we discuss the problem of solving a multiclass classification problem in which one class is not only rare, but a composition of other classes as well. Precisely we present our solution to the Plant Pathology challenge on Kaggle: https://www.kaggle.com/c/plant-pathology-2020-fgvc7/. Our solution uses a composition of the most state-of-the-art deep neural networks, which focus on being both fast and accurate. In addition to that an ancillary deep network is trained, which specifically deals with the rare class. To support this additional network an sophisticated method for upsampling the rare class is used. This network is then braided into the composition of deep neural networks to form an architecture which is able to deal with the rare class while still classifying the other classes correctly. We achieve a score of 96.6%*, on the public leaderboard which is in the top* 41% *of all competitors.*

1. Introduction

If one wants to prevent an outbreak of disease in an apple orchard one either has to consult (or be) an expert for plant diseases, who are unfortunately rare and most of the time overbooked [16, 36]. Experts are needed because detecting and classifying diseases in an apple orchard is very hard and requires a lot of training. Without expertise it is not possible to distinguish between different diseases or evaluate how serious a situation is [36]. But unnoticed diseases can lead to cosmetic unappealing apples and even to ones which are so diseased that they are unsuitable for sale [33]. This will lead to high economic losses and hence one needs to react fast if there is an outbreak of disease. Many experts in pant pathology use a combination of visual features found on the leafs of apple trees and automatic evaluation tools [3]. However are such evaluation tools prune to human error because they need human intervention and are far from automatic [3]. Additionally to that are those tools often very complex and difficult to use [3].

Both human experts and evaluation tools heavily evaluate visual indicators for detecting diseases, so to try to detect diseases solely on images feels natural. But visual indicators for foliar diseases range greatly depending on leaf color, ambient temperature, time of the day, age of the considered apple tree and a whole range of more variables [7, 36]. Because of that is the result sometimes unsatisfactory even if the consulted expert uses additional software and produces no errors [36]. If it was possible to detect foliar diseases only with images, one could prevent disease outbreaks on apple orchards equipped only with a camera. Such a classification would be much faster and cheaper than consulting an expert on plant pathology. Some solutions to plant pathology have emerged in recent time which only rely on computer vision. But most of the solutions do not hold up to the expectation of classifying leafs while being on the apple orchard. Most existing solutions assume that the leafs are evaluated in an laboratory environment and hence to not deal with external influences like different backgrounds, different lighting conditions or other factors which come into play if the leafs should be classified on the apple orchard [3, 26, 23, 10].

Since this problem is unsolved Thapa et al. [36] created an annotated dataset and started a Kaggle challenge (https://www.kaggle.com/tarunpaparaju/plant-pathology-2020-eda-models) to solve the problem of classifying leafs in an natural environment to be used on the go while walking through a apple orchard. This work tries to solve this problem with the provided dataset.

2. Dataset

The dataset is provided by the Plant Pathology and Plant-Microbe Biology Section of the Cornell University [36]. The dataset consists of images of leafs from apple trees under multiple health conditions. The images were taken from different angles, various illumination conditions and several different times of day. The dataset was generated directly from apple orchards. For more details see [36].

The dataset consists of four disjoint classes, so every

sample belongs to only one class. The classes are: *Healthy*, *Rust*, *Scab* and *Multiple diseases*. Every sample is an image of a leaf which belongs to one of the classes. There are 3642 samples in total. 3619 samples are of size 2048×1365. 23 samples have flipped dimensions of 1365×2048. The samples are split into train images and validation images. There are 1821 images in both categories (so it is a split in two halves). The labels for the train images are given. The validation images are unlabeled however, since they are used as performance evaluation in the Kaggle challenge. Figure 1 shows and example of every class.

Class	Frequency	
	Absolute	Relative
Healthy	516	28.34%
Rust	622	34.16%
Scab	592	32.51%
Multiple Diseases	91	5.00%

Table 1: Class distribution of the Train Images.

Since only the classes of the 1821 training images are given merely they can be observed. The class distribution of the dataset is highly imbalanced. While the classes *Healthy*, *Rust* and *Scab* are fairly balanced there is a lack of images of the *Multiple diseases* class. For details see Table 1.

3. Methods

The images of this dataset are all within the same class of "leafs" and hence it is not possible to rely on the high variance of different classes (e.g. differentiating between cats and dogs). The focus needs to be set on the fine grailed characteristics of the different classes. This problem is called fine-grained image classification [2]. This is especially difficult since the *Multiple Diseases* class may contain properties contained in the *Rust* and/or *Scab* classes as well. One additional challenge is that the *Multiple Diseases* class is underrepresented and hence cannot be learned as easy as the other classes. The main challenges of classifying the different leaf conditions are therefore:

- Learn the fine-grailed features of the different leaf conditions.
- Compensate the few samples of the *Multiple Diseases* class.

Firstly some necessary preprocessing steps and image augmentation techniques are presented in section 3.1 to prepare the data for training, enrich the dataset and make it less prone to overfitting. Secondly the general problem of classifying the different leaf types is tackled in section 3.2. After that, in section 3.3, an additional technique is introduced which focuses on the rare *Multiple Diseases* class. Finally, in section 3.4, are the two methods of section 3.2 and section 3.3 combined to form a architecture which deals with the proposed problem in a composed way.

3.1. Preprocessing and Image Augmentation

The train images are once more split into a train- and test-data. 90% (1639 images) of the images are used for training and 10% (182 images) for testing. The datasets are referred to as "train-dataset" and "test-dataset" respectively.

Even simple image augmentation has shown to have a great impact on the performance on deep neural networks [27, 31]. Other approaches to leaf classification usually use affine transformations, simple image rotations and perspective transformations [32, 37]. In this work the augmentation techniques are also mostly simple affine transformations. The augmentations are namely:

- Horizontal Flips.
- Vertical Flips.
- Rotations within a range of 10 degrees in both directions.
- Horizontal shifts in the range of 0%-10% of the total image width in both directions.
- Vertical shifts in the range of 0%-10% of the total image height in both directions.
- Zooms in the range of 90%-110% (so zooming 10% in or out).
- Shearing transformations within a range of 10 degrees (counter-clockwise).
- Darken the image within a range of 50% of the original brightness value.
- Brighten the image within a range of 50% of the original brightness value.
- Box blur with a kernel size of (5×5) to blur the image. [1].

The augmentations are applied randomly when training data is sampled. The test-dataset is not augmented. If empty parts arise in the course of augmentation they are filled by duplicating the nearest pixels until the empty space is filled. All images are normalized by scaling the pixel values to the range of 0-1 $(1/255)$. All images are scaled to a size of (299×299), for the sake of performance. All 23 images of flipped dimension are rotated to have the same dimension as the other images in a preprocessing step.

[1] For more information about box blur see: [4].

(a) Healthy Sample **(b)** Rust Sample **(c)** Scab Sample **(d)** Multiple Diseases Sample

Figure 1: Examples of samples from all classes of the Plant Pathology Dataset.

3.2. Fine-Grailed Classifcation

Recent studies in deep neural network architectures showed great performance increases in image classification, even on fine-grailed inter-class classification problems with low variance, while keeping the computational complexity manageable. Especially variations of Convolutional Neural Networks (CNNs) have seen great progress. Smart advancements in the network architectures moved away from just making the networks *bigger* to making the network architectures *smarter* to achieve great accuracy while having manageable computational complexity.

In [34] Szegedy et al. proposed the famous **Inception Network** which is able to classify images of the same class even if there is a large variation in how the images look like (e.g. a dog which is near the camera and one which is far away). The Inception Network is a 22 layer deep CNN. Szegedy et al. used several very small filters of different sizes on every level to make the network more sparse and hence faster while achieving remarkable performance [34].

He et al. presented the **Residual Neural Network (ResNet)** architecture in [12]. ResNet is a novel architecture which complements the CNN architecture with so called "skip connections" which jump over some layers (this idea is inspired by the pyramidal neurons [29] in the cerebral cortex) [12]. Similar to the Inception Network ResNets exploit the idea of thinner, but deeper networks to be faster than comparable architectures.

In [14] Huang et al. proposed an improved version of ResNet called **Densely Connected Convolutional Network (DenseNet)**. DenseNet develops the idea further that CNNs can be much deeper, faster and more accurate if skip connections are used. Each layer in DenseNet has access to the information of all previous layers, which intensifies the idea of skip layers. This makes it possible for the network to be thinner, since each layer receives the feature maps from all preceding layers [39].

In [13] Howard et al. present an architecture built for fast and accurate classification intended to be used in mobile and embedded vision applications. Hence the name **MobileNets**. This architecture is based on depthwise separable convolutions [6] to speed up computation significally. Additionally to that do MobileNets also have parameters to balance the trade off between speed and accuracy.

Lastly developed Tan and Le a new group of CNNs called **EfficientNets** in [35]. The main contribution of this work is to define a technique on how to scale CNNs compoundly in width, depth and resolution all at once [35]. The base model for EfficientNets is very similar to an improved version of the MobileNet, called MobileNetV2 [30]. EfficientNets achieve both higher accuracy and better efficiency than existing CNNs by a margin [35].

Ensemble learning (in terms of classification) is the idea that a number of classification algorithms aggregated in some way usually perform much better than a single classifier [28]. Ensembles work since having the opinion of multiple learners averages out biases and reduces the variance, since if one aggregates multiple opinions the noise will be much lower than with a single classifier [9, 38]. Also are ensemble methods unlikely to overfit since the training of the individual models is independent from each other and hence are the trained parameters as well [38]. Even though there exist very sophisticated ensembling techniques often times simple averaging over the ouputs of the models is sufficient [21, 25]. In this work the most basic form of ensembling, the simple average $\tilde{y}(\mathbf{x})$, is used. It is defined as:

$$\tilde{y}(\mathbf{x}) = \frac{1}{p}\sum_{i=1}^{p} y_i(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is the input vector, p is the number of the consulted classifiers and $y_i(\mathbf{x})$ is the i-th classifier.

3.3. Detecting The *Multiple Diseases* Class

Since the class *Multiple Diseases* only makes up about 5% of all images it is unpreventably harder to classify than the other classes. Additionally to that it may also contain characteristics which are typically affiliated with samples of the classes *Rust* or *Scab*. For example contains the leaf of the class *Multiple Disease* in Figure 1d several brown spots which are typical for the *Rust* class which can lead to confusion. To reduce the difficulties which arise with the *Multiple*

Disease class an additional classifier is employed in which the problem is reduced to a simpler *Multiple Disease* and *Not Multiple Disease* differentiation.

Since EfficientNets showed the best accuracy (see section 3.2) an additional EfficientNet is employed for this reduced problem.

The biggest problem with training an network on the two defined classes only is that this reduction intensifies the imbalance problem, since now the *Not Multiple Disease* class, which is a compound of the *Healthy*, *Rust* and *Scab* classes, makes up 95% of the train-dataset. To reduce this imbalance oversampling is used. Oversampling is a technique which tries to reduce the problem of imbalanced classes. The most naive approach to oversampling is called random oversampling, which duplicates samples of the minority class (*Multiple Disease*) to create a more balanced situation [11]. This approach increases the chance of overfitting. The more random oversampling is applied the higher the chance of fast overfitting [5].

A much more sophisticated approach is called the **Synthetic Minority Oversampling Technique (SMOTE)**. SMOTE creates new, synthetic, data rather than just oversampling by duplicating existing samples. Even though SMOTE is usually used in structured data, it can be used for images as well [1, 15]. SMOTE chooses a random sample of the minority class and finds its k-nearest neighbour samples (in terms of euclidean distance). Then a random sample of the k neighbour samples is chosen. Then a new synthetic sample is created on a random point between the two samples. So if we want to double the number of samples in our minority class *Multiple Disease* we take each sample $\mathbf{s_i} \in$ *Multiple Disease* and select one of its k-nearest neighbours $\mathbf{n_j}$). Then we calculate the difference vector d between $\mathbf{s_i}$ and $\mathbf{n_j}$ and multiply d with a random number δ between 0 and 1 to get a new sample (vector) s_{new}:

$$\mathbf{s_{new}} = \mathbf{s_i} + (\mathbf{n_j} - \mathbf{s_i}) * \delta, \;\; \delta \in \mathbb{R} : 0 < \delta < 1 \qquad (2)$$

[24]. This is repeated until enough samples of the minority class are generated. Figure 2 shows some examples of samples generated with SMOTE.

3.4. The Combined Model

The used architecture is a concatenation of a "simple average" ensemble of the presented methods in section 3.2 and the classfier speciliazed in detection the *Multiple Disease* class presented in section 3.3. An sketch of the architecture can be seen in figure 3.

The simple-average Ensembler takes the output of each of the five classifiers and averages them (see section 3.2). The output of each classifier is a vector of length four, one value for each class indicating of probable it is that the input is within this class. So the output of the ensembler is logically a vector of length four as well. The *Multiple Disease* detector (section 3.3) outputs a single number in the range $[0 : 1]$ indicating how probable it is, that the given input is within the class of *Multiple Diseases*, the higher the more probable. We call this output $prob_{mult}$. So to combine both outputs the output of the *Multiple Disease* detector is extended to the length four as follows: for every class that is not *Multiple Disease* the value will be $1 - (prob_{mult})$ and for *Multiple Disease* it is simply the output $prob_{mult}$. This can be formulated as:

$$[\frac{1}{p}\sum_{i=1}^{p}(1-\delta)y_i(\mathbf{x})] + \delta \begin{pmatrix} 1 - y_{p+1}(\mathbf{x}) \\ y_{p+1}(\mathbf{x}) \\ 1 - y_{p+1}(\mathbf{x} \\ 1 - y_{p+1}(\mathbf{x})) \end{pmatrix}, \qquad (3)$$

where $y_i(\mathbf{x}) \leq p$ are the classifiers of the ensemble and $y_{p+1}(\mathbf{x})$ is the *Multiple Disease* detector. δ is a meta-parameter to indicate how important the *Multiple Diseases* Detector should be. For this work δ is always set to 0.5. This setup requires the *Multiple Disease* class to be at index 1.

Each of the classifiers in the ensembler have two additional dense layers appended to the base model, to make the models more flexible. The first has 512 units and the sec-

(a) Synthetic image 1 **(b)** Synthetic image 2

Figure 2: Examples of sythetic samples generated with SMOTE.

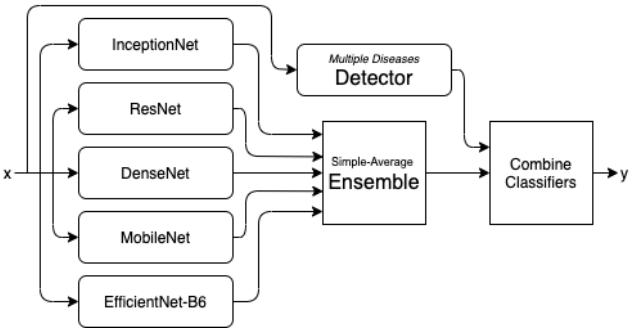

Figure 3: Sketch of the used architecture for solving the Plant Pathology Challenge.

Classifier	F1-Score Per Class			
	Healthy	**Multiple Diseases**	**Rust**	**Scab**
InceptionNet	0.969	0.444	0.960	0.950
ResNet	0.989	0.4	0.951	0.944
DenseNet	0.969	0.285	0.953	0.959
MobileNet	0.989	0.555	0.976	0.951
EfficientNet-B6	0.979	0.695	0.976	0.983
***Multiple Disease* Detector**	0.888	0.991	0.888	0.888
Combined	1.0	0.96	1.0	0.991

Figure 4: The F1-Score for each neural network used for each class. For the *Multiple Disease* Detector the continuous output is discretized (if the score is $>$0.5 it is classified as *Multiple Diseases* and if it is $\leq$ 0.5 as *Not Multiple Diseases*). Since the *Multiple Disease* Detector only differentiates between these two classes, the score of *Multiple Diseases* is listed under *Multiple Diseases* and *Not Multiple Diseases* under every other class to get an uniform look.

ond 256 units. Both dense layers have a ReLU activiation function. In between the two layers is a 30% dropout layer.

Every classifier is loaded with pretrained weights, trained on the ImageNet [8] dataset.

All classifiers in the ensemble are trained with the Adam optimizer, a learning rate of 0.001 and Categorical Cross-Entropy Loss. The *Multiple Disease* detector is trained with Binary Focal Loss [22] with $\alpha = 2$ and $\gamma = 0.25$ and the Adam optimizer with a learning rate of 0.001.

The classifiers of the ensemble have a 4 unit dense layer with a softmax activiation function as output, to get the desired vector of length four. The *Multiple Disease* detector has a 1 unit dense layer with a sigmoid activation function as output, to get the desired output in the range of $[0 : 1]$.

4. Results

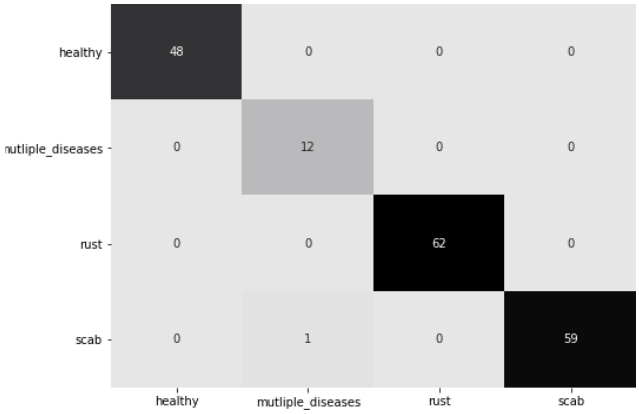

Figure 5: The confusion matrix on the test dataset (182 images).

We trained all models for 200 epochs and a batch size of 6. The samples were randomly sampled and augmented like explained in section 3.1. All models were tained indenpendently of each other and then combined like described in section 3.4.

Observations have shown that the *Multiple Disease* detector sometimes distorts the result if it is unsure if the sample is within the *Multiple Disease* class. Hence it is only considered if the probability value is above the threshold $\epsilon = 0.8$. If the probability value is below the threshold ϵ the output of the detector is not considered (which reduces the model to a simple ensemble classifier).

First of all we look at the performance for each classifier by itself. Table 4 shows the F1-score for each classifier divided up by class. It is obvious that the *Multiple Diseases* Detector plays a big role in detecting the *Multiple Diseases* class, since all other classifiers are rather confused about this class. The other classes however are detected with very high accuracy. The concept for the architecture is verified to be useful if we look at the table: the standard classifiers deal with the well sampled classes without problem but struggle with the rare *Multiple Diseases* class which is compesated by the *Multiple Diseases* detector. The combined result shows that the composition works. The table also shows that the *Multiple Diseases* detector would need more attetion in an improved version of this architecture since it contains the biggest uncertainties in the whole construct, which also explains why the performance increases if an threshold is introduced (like explained in the last paragraph).

Figure 5 shows the resulting confusion matrix on the test data on the combined model. The test data consist of 182 images (48 *Healthy*, 12 *Multiple Diseases*, 62 *Rust* and 60 *Scab*). Only one image was misclassified. This one misclassification confused a *Scab* image as a *Multiple Disease* image. The misclassified image can be seen in figure 6. We assume that this image is falsely classified as *Multiple Disease* because it posses features which could be seen as a mixture of *Rust* (red rectangle) and *Scab* (blue rectangle).

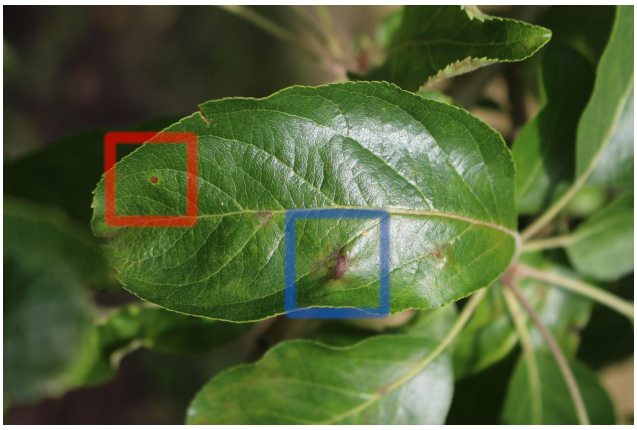

Figure 6: *Rust* image confused as *Multiple Diseases*. Red rectangle shows typical feature for *Rust* class, blue rectangle typical feature for *Scab*.

The trained network architecture achieved a score of 96.6% accuracy on the Kaggle challenge. The accuracy was evaluated on the test dataset of which no labels were given.

5. Discussion

The score of 96.6% on the public Kaggle leaderboard is not all that great. There are a host of reasons why many scored much better than our solution.

Firstly our image size of 299×299 is rather small compared to the leading solutions. Leaders of this competition usually used an image sizes at about 700×700 which enables more fine detection of the features [18, 17, 19]. Secondly is the simple average ensemble rather naive, a weighted sum would lead to better results since not all classifiers will perform equally good (this would involve to train the weights). Some competitors pointed out that the data is flawed, so cleaning of the data could have let to less confusion [20].

All in all does the architecture seem solid and like a sophisticated approach into dealing with rare classes without suffering classification loss on the other classes.

6. Conclusion

In this work we presented an architecture for classifying leafs into multiple classes based solely on image data. We especially tackled the problem of detecting a class which is simultaneously rare and a composition of other existing classes. We did this by training a special classifier which is specialized in detecting whether a sample is of that special rare class. This detector was embedded into a ensemble of classifiers which distinguished between all classes (not only the special rare class). We achieved satisfactory results, if one takes into account that we tried to balance performance and efficiency by scaling the images down highly and only taking network architectures into consideration which are lightweight in terms of computational efficiency.

References

[1] Wajira Abeysinghe, Chih-Cheng Hung, Slim Bechikh, Xiaosong Wang, and Altaf Rattani. Clustering algorithms on imbalanced data using the smote technique for image segmentation. In *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, pages 17–22, 2018.

[2] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015.

[3] Jayme Garcia Arnal Barbedo. An automatic method to detect and measure leaf disease symptoms using digital image processing. *Plant Disease*, 98(12):1709–1716, 2014.

[4] Box blur: `https://en.wikipedia.org/wiki/Box_blur`. — Wikipedia, the free encyclopedia, 2020. [Online; accessed 07.06.2020].

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[7] Pengbo Dai, Xiaofei Liang, Yajing Wang, Mark L Gleason, Rong Zhang, and Guangyu Sun. High humidity and age-dependent fruit susceptibility promote development of trichothecium black spot on apple. *Plant disease*, 103(2):259–267, 2019.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[10] Shiv Ram Dubey and Anand Singh Jalal. Apple disease classification using color, texture and shape features from images. *Signal, Image and Video Processing*, 10(5):819–826, 2016.

[11] Haibo He and Yunqian Ma. *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[15] Brian Alan Johnson, Ryutaro Tateishi, and Nguyen Thanh Hoan. A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees. *International journal of remote sensing*, 34(20):6969–6982, 2013.

[16] Gary JR Judd, Alan L Knight, and Ashraf M El-Sayed. Development of kairomone-based lures and traps targeting spilonota ocellana (lepidoptera: Tortricidae) in apple orchards treated with sex pheromones. *The Canadian Entomologist*, 149(5):662–676, 2017.

[17] Kaggel User: akasharidas: `https://www.kaggle.com/akasharidas/plant-pathology-2020-in-pytorch` . Kaggle, 2020. [Online; accessed 10.06.2020].

[18] Kaggel User: ateplyuk: `ttps://www.kaggle.com/ateplyuk/fork-of-plant-2020-tpu-915e9c` . Kaggle, 2020. [Online; accessed 10.06.2020].

[19] Kaggel User: aziz69: `https://www.kaggle.com/aziz69/efficientnetb7-tpu-tta-top-1` . Kaggle, 2020. [Online; accessed 10.06.2020].

[20] Kaggle User: yelan: `https://www.kaggle.com/c/plant-pathology-2020-fgvc7/discussion/154056` . Kaggle, 2020. [Online; accessed 10.06.2020].

[21] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[23] Bin Liu, Yun Zhang, DongJian He, and Yuxiang Li. Identification of apple leaf diseases based on deep convolutional neural networks. *Symmetry*, 10(1):11, 2018.

[24] Tomasz Maciejewski and Jerzy Stefanowski. Local neighbourhood extension of smote for mining imbalanced data. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 104–111. IEEE, 2011.

[25] Joao Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *Acm computing surveys (csur)*, 45(1):1–40, 2012.

[26] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[27] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

[28] Robi Polikar. Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer, 2012.

[29] Pyramidal cell. — Wikipedia, the free encyclopedia, 2020. [Online; accessed 07.06.2020].

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[31] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

[32] Srdjan Sladojevic, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic. Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, 2016, 2016.

[33] Turner B Sutton, Herb S Aldwinckle, Arthur M Agnello, and James F Walgenbach. *Compendium of apple and pear diseases and pests*. Am Phytopath Society, 2014.

[34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[35] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[36] Ranjita Thapa, Noah Snavely, Serge Belongie, and Awais Khan. The plant pathology 2020 challenge dataset to classify foliar disease of apples, 2020.

[37] Christoph Wick and Frank Puppe. Leaf identification using a deep convolutional neural network. *arXiv preprint arXiv:1712.00967*, 2017.

[38] Zhi-Hua Zhou. Ensemble learning. *Encyclopedia of biometrics*, 1:270–273, 2009.

[39] Yi Zhu and Shawn Newsam. Densenet for dense flow. In *2017 IEEE international conference on image processing (ICIP)*, pages 790–794. IEEE, 2017.