Fruit Tree Disease Detection via Novel Class Incremental Learning Techniques

David Blumenstiel

November 2021

Data Science Master's Thesis

Abstract:

Class incremental learning (CIL) is a current area of interest within machine learning. Machine learning models which can learn to identify new classes incrementally, without complete re-training, would present numerous advantages over conventional machine models. Plant diseases detection offers an interesting use case for CIL, where each different plant introduces new diseased/healthy classes to a model. Two similar CIL techniques for plant disease detection are proposed here and used to identify diseased/healthy plant leaves on the Plant Village dataset. Both techniques combine modular convolutional neural networks (CNNs; sub-networks) into one model, where the sub-networks are introduced incrementally to solve new classification tasks. Both techniques utilized a generic 'other' class in each sub-network for task identification, which contained miscellaneous examples of leaves from plants that were not from classes within their respective sub-networks. Technique-1 used sub-networks with built-in task identification, while technique 2 separated task-identification and disease classification sub-networks into individual components. These techniques were used to incrementally construct models that could classify apple, cherry, and peach diseased/healthy states. These models were approximately 97% accurate, which was not significantly different than a non-CIL, CNN model. The generic 'other' class proved instrumental in task identification, with very little performance loss across model incrementations. These techniques are a promising approach to CIL, with linear computational expense growth across increments.

1. Introduction

1.1. Class Incremental Learning

In a typical supervised machine learning setting, labeled data from all tasks are provided at the time of training (Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & van de Weijer, J., 2021). However, data is not always available for use at the same time. To accommodate this, there is incremental learning, where machine learning algorithms adjust their knowledge in response to new data (Geng and Smith-Miles, 2005); this has any number of real-world applications and advantages, but also contain some significant limitations.

Deep neural networks excel at learning individual tasks but tend to forget old tasks when sequentially trained on new tasks; this phenomenon is often referred to as catastrophic forgetting (van de Ven and Tolias, 2019). Within the realm of computer vision, there is a desire to be able to incrementally add prediction labels to models as new data arrives without needing to completely re-train the model; this is referred to as Class Incremental Learning (CIL; Rebuffi, S., Kolesnikov, A., Sperl, G., & Lampert, C.H., 2017). However, updating model weights to incorporate new labels typically results in either failure to learn new classes, or significant degradation in the performance of the model on the old classes (catastrophic forgetting). Traditionally, if one wants to update the weights of a model to incorporate new labels without

significant biasing, they will typically need to keep around the old dataset for re-training. However, it's not always practical to keep the old data around; it may take up too much storage space, the data may no longer be available, or it may take too long to re-train the model using old data. It would also be convenient to have models that are able to expand their functionality without a complete re-work. Finding a way to add labels to models incrementally without significantly decreasing performance on old tasks is a topic of current research. CIL is closer to human methods of learning, in which we can comprehend new tasks (such as classifying objects) without forgetting how to perform old tasks (Masana et al., 2021).

Numerous CIL techniques have emerged recently. For instance, Leo and Katalia, 2021 employed a method where a classification confidence threshold is used to prime the network for incremental learning and reduce forgetting. Another method, Incremental Classifier and Representation Learning (iCaRL) proposed by Rebuffi et al. 2017, allows for class incremental learning by keeping examples of past data, which helps prevent catastrophic forgetting, and by learning strong classifiers and data representation together. The use of 'exemplars' is controversial however, as one of the main goals of CIL is to eliminate the need to store old datasets. Alternatively, Dai, X., Yin, H., & Jha, N. K. (2020) proposed a grow-and-prune method where a neural network first grows new connections to facilitate integration of new data, then prunes connections based on the magnitude of the weights. A different approach taken by Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., & Kuo, C.J. (2020) involved model consolidation, wherein new models were trained on new data, and then consolidated via a 'double distillation' training objective. Here, the consolidated model is informed by both the new and old models with the help of an auxiliary unlabeled data set.

There are many other examples of CIL, most arising in the past few years. It should be noted that there is some discrepancy concerning the criteria of CIL in the literature. One common theme however is the ability to sequentially learn new tasks, without complete re-training. They all also tackle the issue of catastrophic forgetting either using only a subset of the old data, or none of the old data.

Plant disease detection may provide a good use case for CIL. Plants diseases are typically diagnosed visually, thus it's a reasonable assumption that digital images can contain information pertinent to disease (class) and plant (task) identification. Moreover, there are also publicly available datasets of plants with labeled diseases widely available online. Computer vision models exist to classify plant diseases, but to the author's knowledge, no CIL techniques have been applied to plant disease detection.

1.2. Crop Pathogens and Pests

Pathogens and pests have presumably afflicted human agricultural ambitions since their inception. Oerke (2005) estimated that, in modern times, plant pathogens alone reduce global crop yield by around 16%. The Food and Agricultural Organization of the United Nations further estimates that between 20 and 40% of global crop yield is lost to pests (UNFAO, 2019).

Combating plant pathogens and pests, hereby collectively referred to as ‘diseases’, poses challenges for growers globally.

Diseases, which are common to every cultivated type of fruit tree, can reduce the quantity and quality of fruit produced. Disease is one of the primary causes of lost yield, and determining which diseases afflict any particular tree is crucial when it comes to prevention and treatment (Tian, Y., Yang, G., Wang, Z., Li, E., & Liang, Z., 2019). Typically, diagnoses are done in person, but this can be error prone and costly (if hiring an arborist). Diagnosis via human observation is also time consuming, and not practical for large-scale, in-depth analyses.

Arable land is a finite resource. Also, developing additional arable land for crop production tends to involve the disruption of or removal of natural environments, which has adverse effects on climate and biodiversity when done on a large scale. Thus, ensuring a high crop yield density is one way to help sustainably meet demand, especially as the world population continues to grow, and standards of living improve. Combating plant disease is one task necessary to ensure crop production (both of food and other resources) continues to meet demand. Fortunately, there are many new and promising tools arising to meet this need.

1.3. Disease Detection via Computer Vision

Fruit tree diseases tend to manifest themselves visually on leaves, branches, fruits, or combinations thereof, which makes optical diagnoses the go-to. Computer vision, a field of machine learning where information is extracted from images, has been of interest when it comes to detecting plant disease (Liu and Wang, 2021). Computer vision models are able to detect disease accurately and conveniently, often only requiring a digital image. It’s also possible to scale computer aided disease detection systems up to survey entire orchards at once via aerial imagery.

CNNs in particular are used to great effect, both for diagnosing plant diseases and in other classification problems. There are many instances of computer vision techniques used for fruit tree disease detection. For example, Yadav, S., Sengar, N., Singh, A., Singh, A., & Dutta, M. K. (2021) were able to create CNN models that could correctly distinguish between healthy peach leaves and those affected with peach bacteriosis 98.75% of the time. Liu, B., Zhang, Y., He, D., & Li, Y. (2017) were able to create a CNN that could detect several apple diseases, including mosaic virus, cedar apple rust, apple scab, and Alternaria leaf spot with 97.62% accuracy. Many models have also been created that can detect disease amongst more than one type of plant. For example, Khan, M. A., Akram, T., Sharif, M., Awais, M., Javed, K., Ali, H., & Saba, T. (2018) were able to create a model that could distinguish between many apple and banana diseases. There are also methods which do not involve CNNs, although these are less common now due to the high performance of CNNs. For instance, Habib, Md. T., Majumder, A., Jakaria, A. Z. M., Akter, M., Uddin, M. S., & Ahmed, F. (2020) made a system for detecting papaya disease which used K-means clustering in conjunction with a support-vector machine capable of achieving accuracies over 90%.

Most plant disease detection models tend to only work for one or a few types of plants and would require retraining to detect more diseases or species. It would be advantageous to have a model which could be progressively constructed to incorporate new diseases and species without complete re-training. However, CIL techniques are still quite new, and are still being developed. Here, two new (to the author's knowledge) CIL techniques are presented, and used to detect and classify diseases in fruit trees given digital images of diseased (or healthy) leaves. Models here will utilize these techniques to incrementally construct models to include new species of fruit tree, one at a time, and their accompanying disease classes. Task identification via a novel technique will be a main focus.

2. Methods

2.1. Data Sources:

Models were trained and evaluated on the Plant Village dataset; this dataset contains over 50,000 labeled images of diseased and healthy leaf images from various food crops (Huges and Salathé, 2016). Included in this dataset are several types of fruit tree: apple, peach, cherry, and orange. This dataset was chosen for its simple imagery, ease of access, and size. Limiting the type of objects to solely leaves reduces the complexity, making fruit tree disease modeling more approachable, while remaining practical. Also, because the images are from several different crops, CIL may be approached on a crop-by-crop basis, where each crop may be considered as a new task.

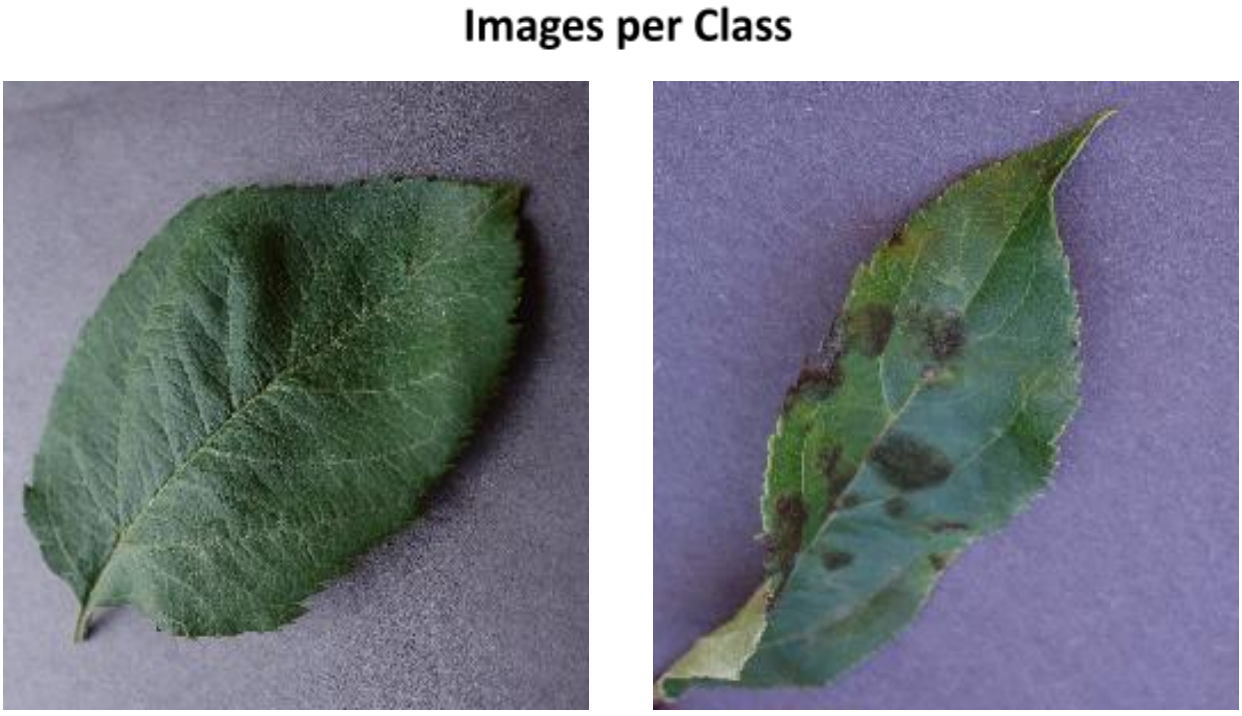


Figure 1. Images of a healthy apple leaf (left) and an apple leaf displaying signs of apple scab disease (right). These are the base images used in modeling prior to augmentation.

Each of the base images contained a singular leaf on a plain (typically gray) background (Figure 1). The images were red-green-blue (RGB), 256 by 256 pixels. Datasets were not balanced, and often contained different proportions of healthy vs diseased leaves (Table 1). The primary species of interest in this dataset were apple (3171), peach (2675), cherry (1732).

Plant	Healthy	Disease 1	Disease 2	Disease 3	Total
Apple	1645	630 (Apple Scab)	621 (Black Rot)	275 (Cedar Apple Rust)	3171
Peach	360	2297 (Bacterial Spot)			2675
Cherry	684	1052			1732

Table 1. Image counts by plant type and disease/healthy. Classes were not well balanced, and there were more apple diseases than peach or cherry.

2.2. Data Preparation

Prior to modeling, 10% of the dataset was withheld for later evaluation purposes; this data was not used for training or validation. A further 20% of the dataset was used for validation during the training phase. The remaining 70% of the data was used for training. Pixel color values, originally between 0 and 255, were rescaled to 0 through 1. Images were then downsized to 224 by 224 pixels. Various augmentations were also implemented on the training and validation images including: vertical and longitudinal shifting, horizontal flipping, rotating, shearing, dimming and brightening, and zooming. Mirroring was used to fill void spaces induced by image manipulation. The intent behind image augmentation was to make more robust models by artificially creating a more varied dataset for training. Avoiding overfitting is a common challenge in machine learning; using image augmentation is a simple way to make data more complex, without finding larger datasets, so models trained on it generalize better with unseen images. Testing images were only rescaled, and not augmented. Data preparation and augmentation were done using the ImageDataGenerator function included in the Keras API for Tensorflow.

2.3. Model Structure

The models implemented here were based on Convolutional Neural Networks (CNNs). However, what distinguishes the model architectures here are the use of multiple separate CNNs within the same model (referred to as ‘sub-networks’). The main objective was to create an architecture that could be incrementally expanded to include additional tasks, each containing new classes, by training new CNN-sub-networks and appending them to prior sub-networks in parallel, negating the need to retrain the prior sub-networks. Two similar modeling techniques, with slightly different architectures were explored here (Figure 2).

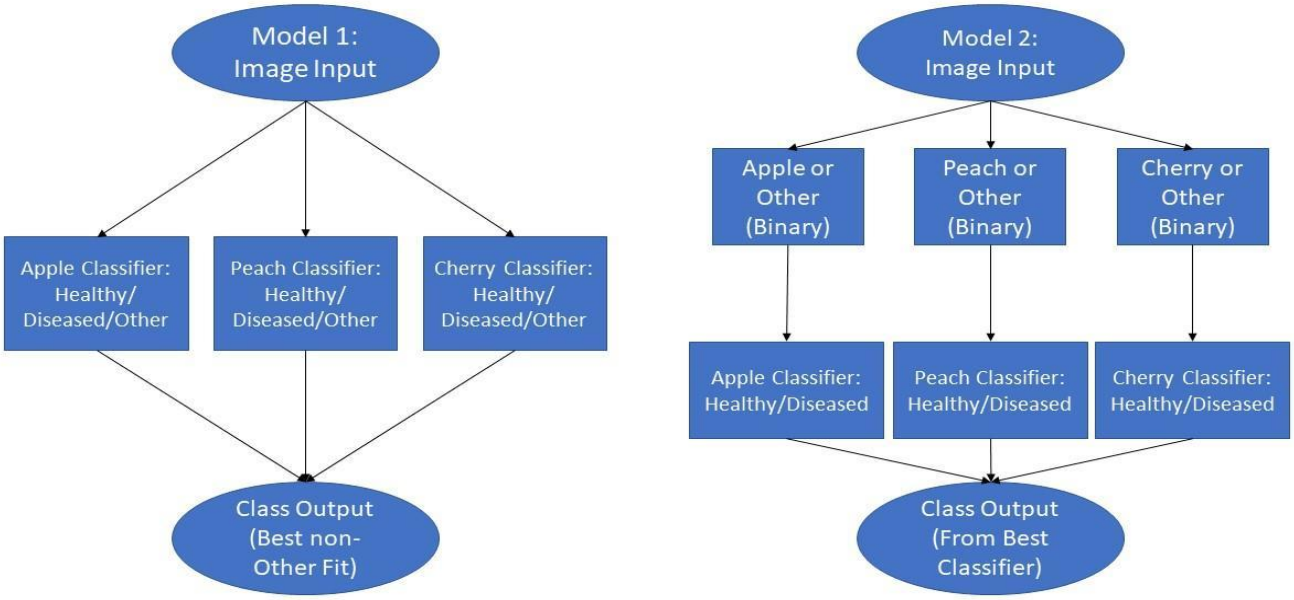


Figure 2. Model architectures: there are two types. Type-1 (left) uses individual CNN sub-network classifiers for each plant. Type-2 (right) uses both binary classifiers (to determine if the image contains the appropriate plant), and a classifier to determine disease. Type-1 returns the best, 'non-other' class fit as the class. Type-2 will first run images through the first binary classification level. This will determine what type of plant the image is, after which the image will then be passed to the sub-network in the second level specific to that plant, which will determine disease.

The first type of modeling technique (referred to as 'type-1') utilized one layer of parallel sub-networks, each one trained to differentiate between diseased leaves/healthy leaves for a single type of plant, and leaves from other plants (in-built task identification). This architecture relied on each sub-network's ability to distinguish between leaves from their task-specific (target) plant, and other types of leaves. Each sub-network was introduced incrementally, each expanding number of classifications the model could make. The sub-network for these type-1 models were trained on a dataset of leaf images from their task-plant, with classes for the various diseased/healthy states, and with a generic 'other' class which consisted of a mixed set of leaves from plants exclusive of the sub-network's task plant. The 'other' class was used for task identification: detecting the appropriate sub-network for any given image. As an image was run through all sub-networks in the model, the sub-network which yielded the lowest probability of the image belonging to its 'other' class was identified as the on-task sub-network. This sub-network would then output the (non-'other') class which had the maximum probability; the final output for the model. As such, each sub-network was capable of task identification, and disease/healthy classification.

The second type of modeling technique (type-2) was similar to the first in that it utilized parallel CNN-sub-networks and relied on an 'other' class for task identification, but consisted of

two separate layers. The first layer consisted of binary classifiers which were capable of distinguishing between a task-specific plant, and the generic 'other' class, similar to those in model type-1. However, this first layer did not distinguish between diseased and healthy leaves (both were used in training); it was only for task identification. The second layer consisted of more traditional multi-class classifiers that were capable of distinguishing between healthy and diseased leaves from a single type of plant. For a given image, the first layer would determine the task, after which the image would be passed to the task-appropriate second layer sub-network. New classes could be introduced incrementally by adding additional sub-networks for each type of new plant; one in each layer.

Both types of models used a set of generic plant leaves (these were not all fruit tree leaves) for sub-networks to distinguish between their tree's leaves, and leaves from other trees. The hypothesis was that leaves of a specific plant would be more similar to one-another than the generic mix of leaves from other plants, and that leaves of other plants would be more similar to the generic mix of leaves than the task-specific leaf for any sub-network. It was thought that either alone or in conjunction, the sub-networks would be able to use this phenomenon for task identification; a major obstacle for CIL.

Furthermore, there were two types of data-selections used for the 'other' classes. The first type included images from every class aside from those from a specific sub-network's target plant. This implies that each sub-network may have had the ability to directly tell when an image was off-target via prior examples. The second type of data-selection for the 'other' classes excluded images from target-plants across all sub-networks. These sub-networks relied more on the ability of their sibling sub-networks to recognize their on-target images, because they had no direct examples of off-target images, and for off-target images to generalize to the 'other' class better than disease/healthy classes for the wrong plant. These different types of 'other' classes will be referred to as 'inclusive' and 'exclusive' going forward.

2.4. Training and Sub-Network Structure

For simplicity, ease of training, and comparative purposes, the architecture of the sub-networks within each of the models were kept similar. Transfer learning was used extensively: most of each sub-network consisted of the MobileNetV2 architecture and weights, without the top layer included. MobileNetV2, designed for object detection on mobile devices, was chosen as it has a simple architecture that boasts high accuracy, and is computationally inexpensive compared to other state-of-the-art CNN architectures (Sandler et al., 2018). Keeping the pre-trained weights of the base layers reduced model training time and improved performance. The large dataset (ImageNet) MobileNetV2 was trained on gave the base layers a comprehensive understanding of basic features (lines, colors, etc.).

The top portion of the model started with flattening, followed by a dense layer with 256 neurons using ReLU activation, after which there was batch normalization, followed by 50%

dropout, then an output dense layer with the same number of neurons as classes (including 'other'), using the softmax activation function. The sub-networks were trained over approximately 25 epochs. Checkpointing was used, and the sub-network with the best validation accuracy at any epoch during training was often used in the final model. Learning rate reduction was also carried out when sub-network progress stalled across several epochs. The Keras API for Tensorflow was used to construct and train the models.

Each sub-network was trained individually, using its own dataset subset. The data for each sub-network within model type-1 consisted of one type of plant (e.g. apple) with one class for each diseased/healthy state, and an additional 'other' class. The datasets that the first layer of sub-networks in model type-2 were trained on contained all images of the pertinent plants (diseased and healthy together) grouped into one class, and a separate 'other' class. The datasets for the second layer in model type-2 were trained on data from one type of plant, with diseased and healthy separated into their own classes. There were three types of plant included in each model, with three accompanying (sets of 2, for type-2) sub-networks: apple, cherry, and peach.

Training was done locally. Tensorflow GPU support was utilized to facilitate training. Training was performed using a GeForce GTX 1050, 2GB graphics card (NVIDIA Corporation, 2016) and an Intel Core i5-7300HQ processor (Intel Corporation, 2017).

2.5. Model Evaluation

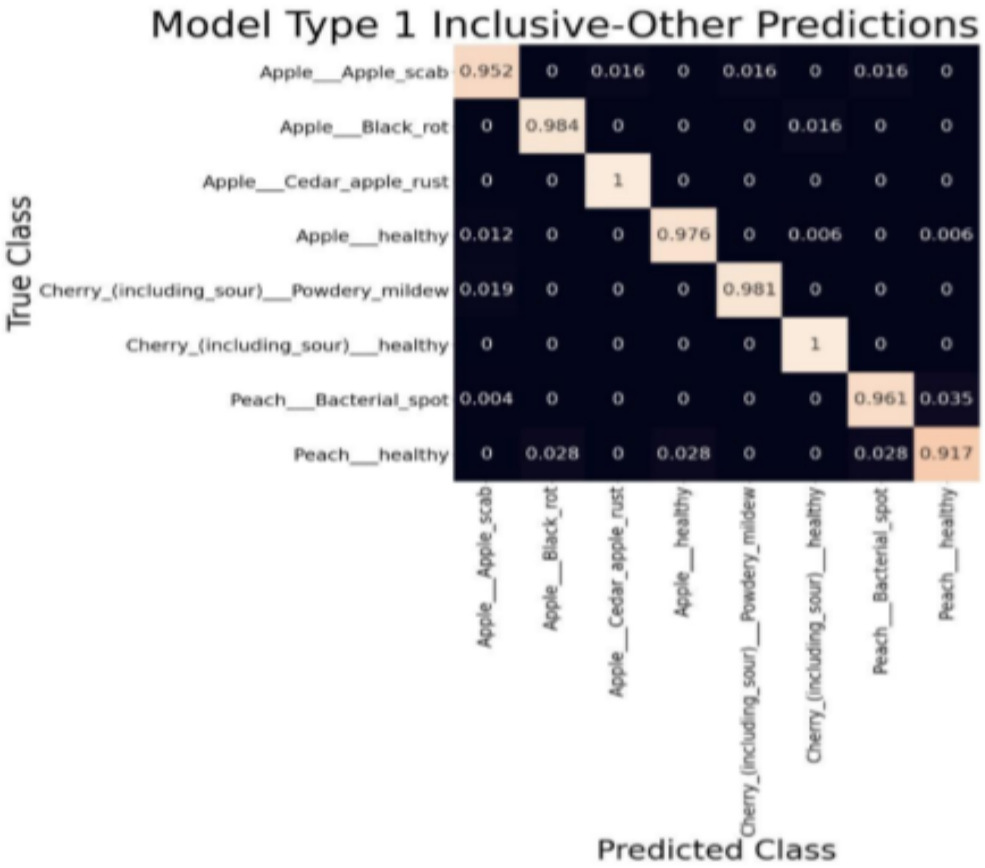
Models were evaluated on holdout (test) sets and compared to one-another using categorical accuracy. Each of the completed models, with three tasks (apple, cherry, peach) were evaluated on holdout sets containing images of apples, cherries, and peaches. A more traditional (non-CIL) combination model was also created for comparison. This traditional model was trained on apple, cherry, and peach diseases/healthy classes all at once; performance on this model will serve as a baseline.

Class incremental accuracy between model iterations (as tasks were added) was also determined by making predictions on a single plant as each task was added, starting with the task specific plant for the first iteration. E.g., apple images were tested on a model with only one sub-network (apples), then on a model with two (apples and cherries), then the full model (apples, cherries, and peaches).

3. Results and Discussion

3.1. Model Type 1 Results

Models constructed with technique type-1, which utilized the output from several different sub-networks, were able to effectively distinguish between tasks (different plants) and plant diseases/healthy classes with a high degree of accuracy; this was for both inclusive and exclusive 'other' models. The type-1 inclusive 'other' model had an accuracy of 97.2% (95%



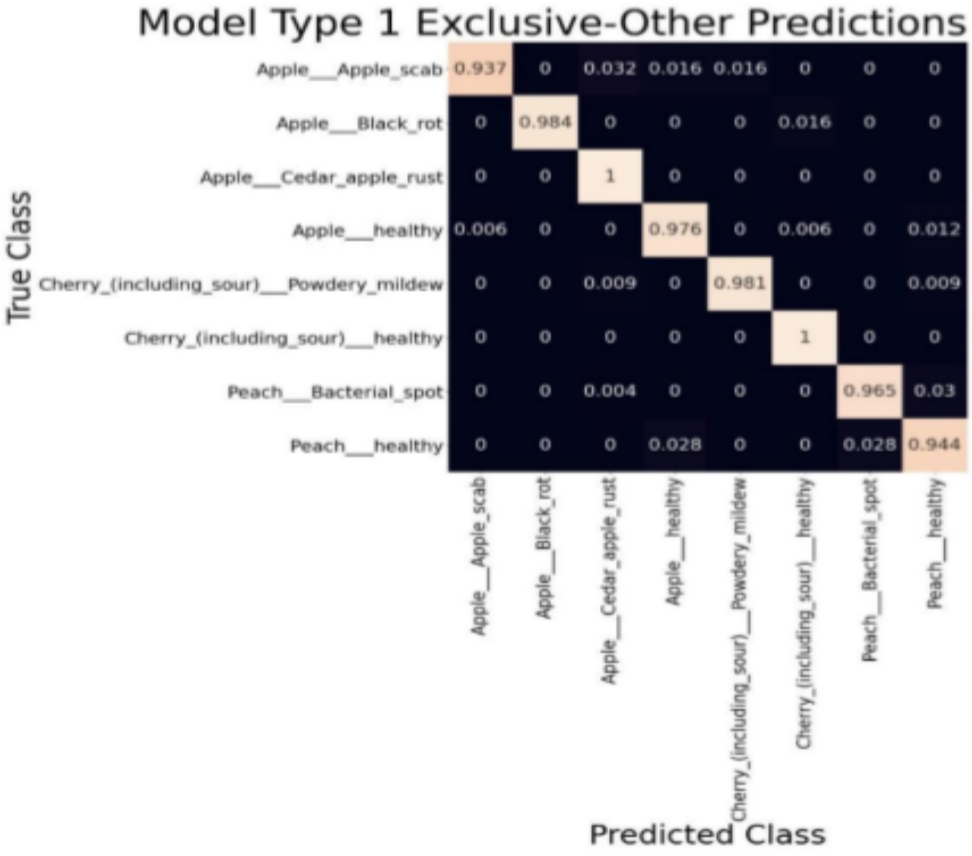


Figure 3.
Proportional Confusion matrices for Type-1 models; both inclusive and exclusive 'other' classes. Numbers within each box are the proportion of each image class predicted for each possible prediction (recall).

Overall predictive accuracy for both types are similar. Both are approximately 97% accurate.

The test set here was completely held out from training and contains 10% of the appropriate data.

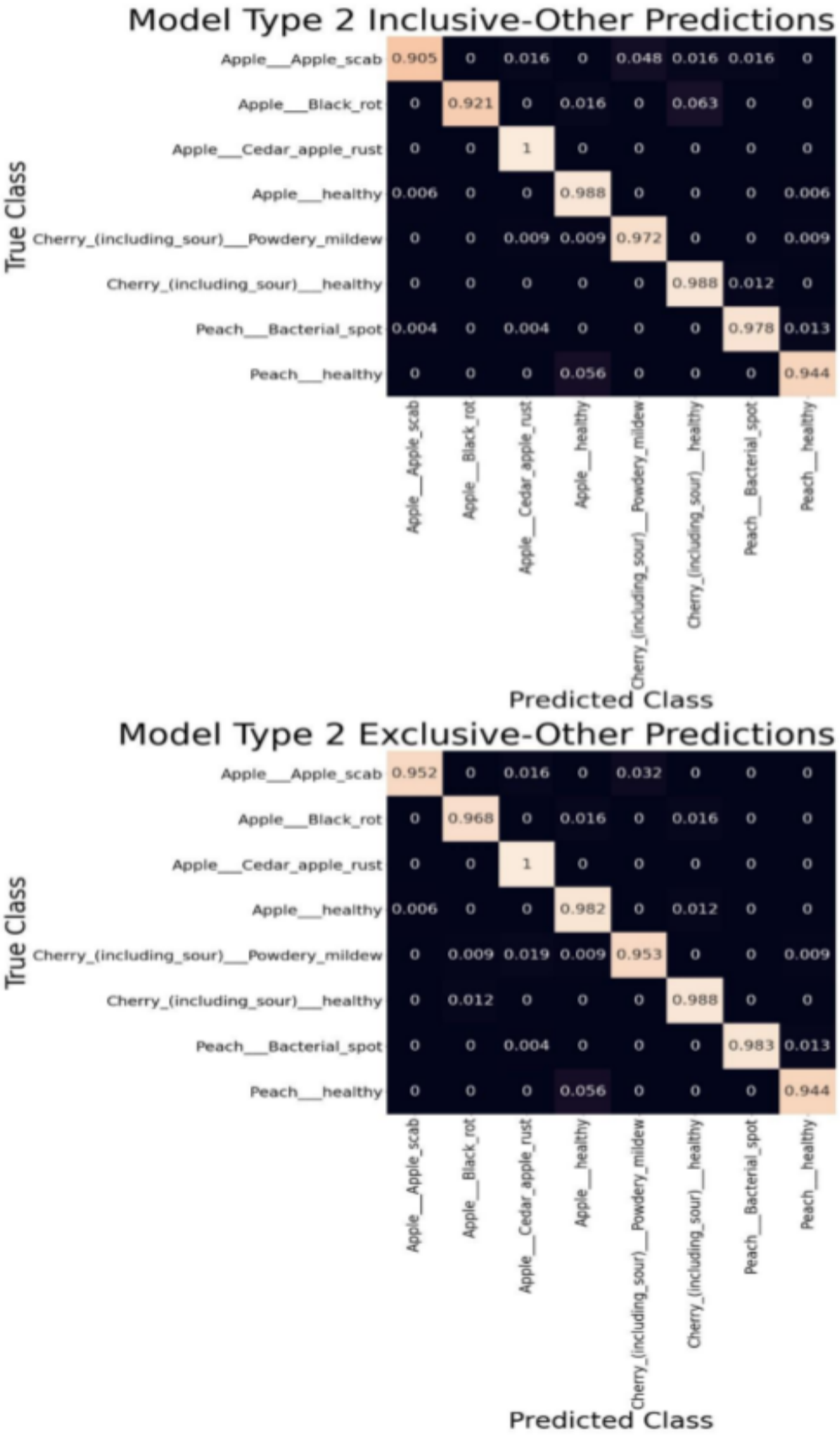


Figure 4.
Proportional Confusion matrices for Type-2 models; both inclusive and exclusive 'other' classes. Numbers within each box are the proportion of each image class ; redicted for each possible prediction (recall).

Overall predictive accuracy for both types are similar. Both are approximately ⅟7% accurate.

The test set here was completely held out from training and contains 10% of the appropriate data.

CI: 96.0 – 98.3%), while the exclusive 'other' model had an accuracy of 97.3% (96.2 – 98.4%). Class imbalances were handled similarly well except for one class, healthy peaches; F1-scores for all other classes were above 0.90.

The 'other' class proved highly instrumental in discriminating between tasks. Given images from target plants, the on-target sub-networks tended to give lower probabilities to the 'other' class than the off-target sub-networks. Thus, the task could be predicted for any given image by selecting the sub-network that suggested the lowest probability of the 'other' class. With the correct task, the disease-class could accurately be determined by picking the largest 'non-other' class from the on-target sub-network. There was no significant difference in the overall accuracy between models with inclusive or exclusive 'other' classes. It appears there may have been a tradeoff between an on-target sub-network's ability to recognize on-target images, and between how well off-target models could recognize off-target images. Overall, it seems that a ('inclusive') model whose sub-networks' 'other' classes were trained with data inclusive of their siblings' target plants were able to better recognize off-target images, but generally weren't as good as recognizing on-target images (their 'other' probability wasn't as low), and vice-versa for exclusive 'other' class models.

3.2. Model Type 2 Results

Models constructed using technique type-2 also performed very well. The first layer of sub-networks was able to successfully identify the correct task for images. The type-2, 'other' inclusive model had an overall accuracy of 96.9% (95% CI: 95.7 - 98.4%), while the 'other' exclusive model was 97.4% (0.963 - 0.985%) accurate (Figure 4).

As with the type-1 technique, the use of a generic 'other' class for task identification in the type-2 technique proved successful. Even when individual first-layer (task identification) sub-networks made misclassifications, the other first-layer sub-networks were often able to correct for that with their predictions. E.g., an apple image is run, but the apple task-identifier sub-network predicts 'non-apple'; the cherry and peach task-identifiers predict 'non-cherry' and 'non-peach' with a higher certainty than the apple task-identifier's prediction. Also similarly to the type-1 models, the 'exclusive' model sometimes had better on-target recognition at the expense of off-target recognition when compared to the 'inclusive' model.

3.3. Model Comparisons

Both modeling techniques were highly accurate, and not significantly different than a traditional CNN model trained on apples, peaches, and cherries all at the same time (non-incremental; table 2). The traditional model was 97.3% accurate, both type-1 inclusive and exclusive models were 97.2% and 97.3% accurate respectively, while the type-2 inclusive and exclusive models were 96.9% and 97.4% accurate respectively. Overall, none of the models were significantly different from one-another in terms of accuracy (at a significance level of

Model	Accuracy	95% Confidence Interval
Type-1 (‘other’-inclusive)	0.972	0.960 - 0.983
Type-1 (‘other’-exclusive)	0.973	0.962 - 0.984
Type-2 (‘other’-inclusive)	0.969	0.957 - 0.981
Type-2 (‘other’-exclusive)	0.974	0.963 - 0.985
Traditional	0.973	0.963 - 0.985

Table 2. Model accuracy comparisons. No model performed significantly differently on the same test set. Confidence intervals were calculated at a significance of 0.05 (95% confidence).

0.05). This offers evidence that these modeling techniques are competitive, in terms of accuracy, when compared to traditional CNN models.

3.4. Incremental Accuracy

Both the type-1 and type-2 exclusive and inclusive ‘other’ models were evaluated incrementally. This was done by using only one type of plant images at a time, while incrementally adding in sub-networks designed for other plants. E.g., apple images were first tested on a model with an apple-specific sub-network, then on a model with both apple and cherry sub-networks, then on a full model with apple, cherry, and peach sub-networks. This allowed for an examination into how much accuracy deteriorated as additional tasks were introduced incrementally. This was done for apples, cherries, and peaches using both inclusive and exclusive ‘other’ models of type-1 and type-2 techniques; the results are detailed in table 3.

Accuracy did tend to drop as additional tasks capabilities were introduced to the models. However, performance losses were not dramatic, with additional tasks only resulting in only one or two misclassifications per class, if any (there were often none). Incremental accuracy only dropped significantly in three instances: cherry for type-2 exclusive, cherry and apple for type-2 inclusive (Table 3). This indicates that these techniques avoided catastrophic forgetting; prior knowledge was retained as additional tasks were introduced. Perhaps just as importantly, that knowledge also remained well utilized. The ‘other’ class, whether inclusive or exclusive, has during these experiments proven to be effective at task identification in a CIL setting.

Type-1 Inclusive			
Plant	**Increment 1**	**Increment 2**	**Increment 3**
Apple	0.991 (0.980 - 1.00)	0.981 (0.966 - 0.996)	0.975 (0.958 - 0.992)
Cherry	1.00 (1.00 – 1.00)	0.990 (0.975 - 1.00)	0.990 (0.975 - 1.00)
Peach	0.966 (0.944 - 0.988)	0.955 (0.93 - 0.98)	0.955 (0.93 - 0.98)
Type-1 Exclusive			
Apple	0.984 (0.971 - 0.998)	0.978 (0.962 - 0.994)	0.975 (0.958 - 0.992)
Cherry	1.000 (1.00 – 1.00)	0.990 (0.975 - 1.00)	0.990 (0.975 - 1.00)
Peach	0.97 (0.949 - 0.99)	0.962 (0.940 - 0.985)	0.962 (0.940 - 0.985)
Type-2 Inclusive			
Apple	0.991 (0.98 - 1.00)	0.966 (0.945 - 0.986)	0.959 (0.938 - 0.981)
Cherry	1.000 (1.00 – 1.00)	0.979 (0.959 - 0.999)	0.979 (0.959 - 0.999)
Peach	0.989 (0.976 - 1.00)	0.974 (0.954 - 0.993)	0.974 (0.954 - 0.993)
Type-2 Exclusive			
Apple	0.991 (0.98 - 1.00)	0.975 (0.958 - 0.99)	0.975 (0.958 - 0.992)
Cherry	1.000 (1.00 – 1.00)	0.969 (0.944 - 0.993)	0.969 (0.944 - 0.993)
Peach	0.989 (0.976 - 1.00)	0.977 (0.96 - 0.995)	0.977 (0.96 - 0.995)

Table 3. Incremental accuracy for each technique. For each increment and plant, an overall accuracy score is given with a 95% confidence interval in parenthesis. During each increment, an additional task capability is added to the model, but the testing data remains the same (a singular plant). Increment 1 only contains the sub-network for the target plant, while off-target sub-networks are added for increments 2 and 3.

3.5. Computational Expense

Prediction and training times for these models increase linearly with the number of tasks because more sub-networks are added for each new set of tasks; one per task for modeling technique type-1, two for modeling technique type-2. Whether or not this is becomes a problem depends on the circumstances, but more tasks mean more computational expense. Prediction and training times are also dependent on the individual designs on the sub-networks, and the number of images used to test and train them. During these experiments, there was only one principal design for sub-networks that was used for the final models, modified only at the last dense layer

(which needed the same number of neurons as the number of classes). It would be possible to use different structures for each sub-network, and even tailor sub-networks to their particular tasks.

It should also be mentioned that during these analyses, type-2 models run each image through all second layer sub-networks. While not implemented here, it would be possible to limit the amount of second layer sub-networks used for type-2 models to just one sub-network per image. This could be accomplished by only sending images to the appropriate second layer sub-networks as identified by the first layer sub-networks. This would reduce the prediction time by almost half. Theoretical computational expenses (training and prediction) for both techniques could be represented by the equations below.

Model Type-1

$$Computational\ Expense = \sum_{i=1}^{n} Si$$

Model Type-2

$$Computational\ Expense = L2_{Sx} + \sum_{i=1}^{n} L1_{Si}$$

Where:

L1 = Layer 1 L2 = Layer 2 n = Number of Tasks S = Sub-Network x = Task Appropriate

4. Conclusions

The techniques presented here performed better than expected, with the final three-task (apple, peach, and cherry) CIL models performing no worse than a traditional CNN model trained on all classes at once (non-CIL). The 'other' class proved instrumental in task identification, with off-target images consistently generalizing closer to the 'other' class within off-target models than on-target images did within on-target models. Further experimentation with these techniques, such as increasing the number of incremental tasks, or changing the composition of the 'other' class would be desirable. However, the techniques presented here offer a promising route to continuous learning models with linear computational cost expanse.

These methods are useful for incrementally adding more tasks for models designed with these methods in mind but have limitations when it comes to combining existing models incrementally. The type-1 modeling method described in this paper needs task specific sub-networks to be designed with an 'other' class representing generic off-target images in mind, which would preclude adapting pre-existing models without re-training. The type-2 modeling technique would likely be able to use pre-existing models in the second layer of sub-networks but would require the training of additional sub-networks for task discrimination in the first

layer. The nature of these techniques limits their ability to utilize pre-existing models as they are, and would be of more use for solving new problems where models have yet to be created.

It's unclear how these types of models might function with significantly different tasks, e.g. differentiating between both plant diseases and types of cars. Theoretically, the sub-networks should still recognize their on-target images better than they would off-target images, but neural-network behavior tends to be less consistent when dealing with less-recognizable (or unrepresented) images. Problems with off-target recognition might be solved with a more representative other class (containing examples of all relevant image types), but this would need to be planned in advance in order to properly train all sub-networks to recognize off-target images.

4.0. References

Dai, X., Yin, H., & Jha, N. K. (2020). Incremental Learning Using a Grow-and-Prune Paradigm with Efficient Neural Networks. *IEEE Transactions on Emerging Topics in Computing*, 1–1. https://doi.org/10.1109/tetc.2020.3037052

The Food and Agriculture Organization of the United Nations. (2019, April 3). *New standards to curb the global spread of plant pests and diseases*. Food and Agriculture Organization of the United Nations. https://www.fao.org/news/story/en/item/1187738/icode/

Geng X., Smith-Miles K. (2015). Incremental Learning. In: Li S.Z., Jain A.K. (eds) *Encyclopedia of Biometrics*. Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7488-4_304

Habib, Md. T., Majumder, A., Jakaria, A. Z. M., Akter, M., Uddin, M. S., & Ahmed, F. (2020). Machine vision based papaya disease recognition. *Journal of King Saud University - Computer and Information Sciences*, *3*, 300–309. https://doi.org/10.1016/j.jksuci.2018.06.006

Hughes, D. P., & Salathe, M. (2016). An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv [cs.CY]*. Opgehaal van http://arxiv.org/abs/1511.08060

Intel Corporation. (2017). Intel Core i5-7300HQ @ 2.50 GHz CPU. [Apparatus and software]. https://ark.intel.com/content/www/us/en/ark/products/97456/intel-core-i5-7300hq-processor-6m-cache-up-to-3-50-ghz.html

Khan, M. A., Akram, T., Sharif, M., Awais, M., Javed, K., Ali, H., & Saba, T. (2018). CCDF: Automatic system for segmentation and recognition of fruit crops diseases based on correlation coefficient and deep CNN features. *Computers and Electronics in Agriculture*, 220–236. https://doi.org/10.1016/j.compag.2018.10.013

Leo, J., & Kalita, J. (2021). Incremental Deep Neural Network Learning Using Classification Confidence Thresholding. *IEEE Transactions on Neural Networks and Learning Systems*, 1–11. https://doi.org/10.1109/tnnls.2021.3087104

Liu, B., Zhang, Y., He, D., & Li, Y. (2017). Identification of Apple Leaf Diseases Based on Deep Convolutional Neural Networks. *Symmetry*, *1*, 11. https://doi.org/10.3390/sym10010011

Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: a review. *Plant Methods*, *1*. https://doi.org/10.1186/s13007-021-00722-9

Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., & van de Weijer, J. (2021). Class-incremental learning: survey and performance evaluation on image classification. *arXiv [cs.LG]*. Opgehaal van http://arxiv.org/abs/2010.15277

Mittal, S., Galesso, S., & Brox, T. (2021). Essentials for Class Incremental Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. https://doi.org/Sudhanshu Mittal. https://arxiv.org/abs/2102.09517v1

NVIDIA Corporation. (2016). GeForce GTX 1050 2GB GPU. [Apparatus and software]. https://www.nvidia.com/en-in/geforce/products/10series/geforce-gtx-1050/

OERKE, E.-C. (2005). Crop losses to pests. *The Journal of Agricultural Science*, *1*, 31–43. https://doi.org/10.1017/s0021859605005708

Rebuffi, S., Kolesnikov, A., Sperl, G., & Lampert, C.H. (2017). iCaRL: Incremental Classifier and Representation Learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5533-5542.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *arXiv [cs.CV]*. Opgehaal van http://arxiv.org/abs/1801.04381

Tian, Y., Yang, G., Wang, Z., Li, E., & Liang, Z. (2019). Detection of Apple Lesions in Orchards Based on Deep Learning Methods of CycleGAN and YOLOV3-Dense. *Journal of Sensors*, 1–13. https://doi.org/10.1155/2019/7630926

van de Ven, G. M., & Tolias, A. S. (2019). Three scenarios for continual learning. *ArXiv*. https://doi.org/abs/1904.07734

Yadav, S., Sengar, N., Singh, A., Singh, A., & Dutta, M. K. (2021). Identification of disease using deep learning and evaluation of bacteriosis in peach leaf. *Ecological Informatics*, 101247. https://doi.org/10.1016/j.ecoinf.202 1.101247

Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., & Kuo, C.J. (2020). Class-incremental Learning via Deep Model Consolidation. *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1120-1129.