Mixtures of Lightweight Deep Convolutional Neural Networks: Applied to Agricultural Robotics

Chris McCool, Tristan Perez, and Ben Upcroft

***Abstract*—We propose a novel approach for training deep convolutional neural networks (DCNNs) that allows us to tradeoff complexity and accuracy to learn lightweight models suitable for robotic platforms such as AgBot II (which performs automated weed management). Our approach consists of three stages, the first is to adapt a pre-trained model to the task at hand. This provides state-of-the-art performance but at the cost of high computational complexity resulting in a low frame rate of just 0.12 frames per second (fps). Second, we use the adapted model and employ model compression techniques to learn a lightweight DCNN that is less accurate but has two orders of magnitude fewer parameters. Third, K lightweight models are combined as a mixture model to further enhance the performance of the lightweight models. Applied to the challenging task of weed segmentation, we improve the accuracy from 85.9%, using a traditional approach, to 93.9% by adapting a complicated pre-trained DCNN with 25M parameters (Inception-v3). The downside to this adapted model, Adapted-IV3, is that it can only process 0.12 fps. To make this approach fast while still retaining accuracy, we learn lightweight DCNNs which when combined can achieve accuracy greater than 90% while using considerably fewer parameters capable of processing between 1.07 and 1.83 fps, up to an order of magnitude faster and up to an order of magnitude fewer parameters.**

***Index Terms*—Agricultural automation, computer vision for automation, recognition.**

I. Introduction

AGRICULTURAL robotics is rapidly gaining interest as shown by the advent of weed management robots such as AgBot II [1] and harvesting platforms such as Harvey [2], see Fig. 1. Robotic vision algorithms that allows them to understand the diverse environments in which they operate are key for the operation of these robots. For instance, AgBot II uses vision to detect and classify weeds at different stages of growth. whereas harvesting robots detect and segment crop, which are often occluded or are similar in colour to the foliage (green on green).

Fig. 1. Top is an image of AgBot II which performs weed management and bottom is Harvey a robotic sweet pepper harvester.

The current approach to performing weed segmentation is to combine a set of shape and pixel statistic features. Haug *et al.* [3] proposed to combine shape features such as the area and length of the skeleton with pixel-based statistics including the mean, median, skew and kurtosis. A random forest classifier was then used to perform classification. This approach requires the definition of features by the researcher, in the hope that they are robust to varying conditions such as differences in stages of growth, illumination and viewpoint.

The advent of deep learning [4], and in particular deep convolutional neural networks (DCNNs), has opened the potential to jointly learn the most appropriate features and classifiers from data. This has the potential to obviate the need to hand craft the best features for a particular task. However, this has yet to receive widespread acceptance and use for challenging agricultural robotic tasks. One of the reasons for this has been the difficulty in training a DCNN from limited data and that many state-of-the-art networks are large making their deployment on resource limited robotic platforms difficult. For instance,

Manuscript received September 10, 2016; revised December 15, 2016; accepted January 19, 2017. Date of publication February 9, 2017; date of current version March 10, 2017. This paper was recommended for publication by Associate Editor H. Liu and Editor J. Kosecka upon evaluation of the reviewers' comments. This work was supported by the Department of Agriculture and Fisheries under the Strategic Investment in Farm Robotics Program and the Australian Research Council Centre of Excellence for Robotic Vision under Project CE140100016.

The authors are with the Queensland University of Technology, Brisbane, QLD 4000, Australia (e-mail: c.mccool@qut.edu.au; tristan.perez@qut.edu.au; ben.upcroft@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LRA.2017.2667039

even the relatively small Inception-v3 model consists of 25M parameters which is small compared to the 180M parameters for VGG [5].

In this paper, we propose a novel approach for training deep convolutional neural networks (DCNNs) that allows us to tradeoff complexity (e.g., memory size and speed) with accuracy. This is achieved through a three-step process. First, we adapt a pre-trained model (Inception-v3 [5]) to the task at hand which leads to state-of-the-art performance. However, as the pre-trained model has been derived for general object classification, using 1 million (1M) images to classify 1000 objects, the adapted model (Adapted-IV3) is complicated consisting of 25M parameters. This makes it relatively slow and so in the second step we use the adapted model to teach much smaller, or lightweight, DCNNs that can have orders of magnitude fewer parameters. Model compression and "distillation" techniques are used to achieve this. In the third step, inspired by [6], we combine a set of K-lightweight models as a mixture model to further enhance the performance of the lightweight models—this has a linear relationship between the number of models and the resultant model complexity.

This approach leads to impressive results for weed segmentation—a critical task for agricultural robots such as AgBot II. The Adapted-IV3 model provides state-of-the-art performance, improving accuracy from 85.9% [7] to 93.9%. However, this model is only able to run at 0.12 frames per second (fps). To make this approach scalable (low memory requirements and fast) while still being accurate, we demonstrate that $K = 4$ lightweight DCNNs can be learnt and then combined as a mixture model to achieve an accuracy of 90.3%. This model uses just $1/25$th the number of parameters of the Adapted-IV3 model and improves the frame rate by an order of magnitude to 1.83 fps.

In the next section, we describe the previous literature and highlight the myriad of boutique features that have previously been proposed. Section III then details the proposed methods and in Section IV we present the experimental results. We then discuss these results and in Section V we summarise our contributions.

II. Literature Review

Weed and crop classification has been explored in robotic vision literature for more than three decades [3], [8]–[11]. In 1986, Shearer [8] proposed the use of pixel-level co-occurrence features to perform plant classification from camera images. In 1998, Zwiggelaar [9] reviewed the spectral properties and their potential use for weed/crop discrimination. A downside of this approach is the need for expensive specialised hyper-spectral cameras.

More recently, researchers have explored methods to classify weeds and plants using shape features. In [11], area, compactness, and major axis were tested as features to distinguish between crop and weed. Area was found to be one of the most effective features, however, in their work the crop had grown to a size much larger than the surrounding weeds. Lin [10] found that utilising seven shape features (area to length, compactness, elongation, aspect ratio, logarithm of height to width, ratio of perimeter to broadness, and ratio of length to perimeter) provided the best classification accuracy for crop/weed classification.

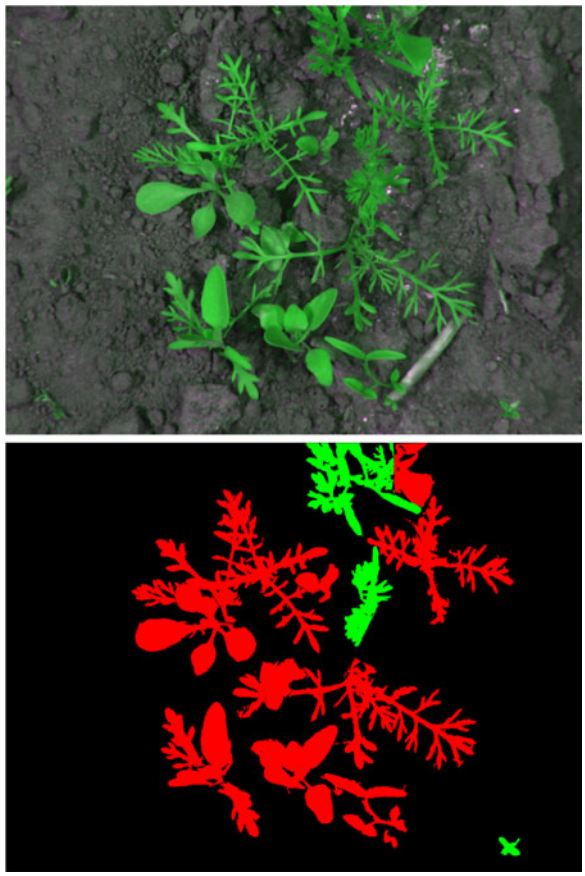

Fig. 2. An example of the challenging weed vs crop segmentation problem. Top is the original image and bottom is the associated ground truth where green represents the crop and red represents the weed.

In 2014, Haug *et al.* [3] proposed to combine shape features with pixel intensity statistics. This combination of features achieved impressive accuracy for weed/crop classification on the publicly available weed/crop dataset [7]. An example image from this challenging dataset is provided in Fig. 2.

This idea of combining features was further extended by Hall *et al.* [12] in 2015. Hall *et al.* proposed to combine traditional features (such as shape and pixel statistics) with features extracted from a DCNN. A pre-trained DCNN was used as a feature extractor and it was combined with the shape and pixel intensity statistics to train random forest classifiers. This led to impressive improvements in accuracy for plant (leaf) classification. Despite the gains demonstrated by Hall *et al.* such an approach has yet to be applied to agricultural robotics. One of the reasons for this is that the network used to extract the DCNN features, AlexNet, is quite complicated, consisting of 60M parameters.

Each solution described above follows a similar pipeline consisting of: (i) empirically finding the best features and (ii) training an appropriate classifier with these features. A recent trend in robotics and computer vision has been to replace this pipeline by *learning* the features from data so that the feature extraction

and classification process can be *jointly learnt and optimised*. Below we describe some of the prior work in feature learning and its relevance for robotics followed by a brief review of recent advances in deep learning.

A. Learning Features

Learning or selecting features has a rich history. One of the first practical examples of this was the Viola and Jones detector for face detection [13] which selected and combined the most appropriate features, from a very large pool of potential features, using boosting [14]. More recently, deep learning has provided a considerable advance in this area by using the raw data (image) as input and learning features as well as the classifier all in the same framework. This led to famous results for general object classification (e.g. cars vs ships) [15] and has been a hot trend in computer vision and machine learning over the last few years [4].

Because of their success, deep learning approaches have been used increasingly in robotics applications. An early example of this, for agricultural robotics, is the sparse auto-encoder features of Hung *et al.* [16] for almond detection. Hung *et al.* learnt efficient auto-encoder features by stacking several layers together, this was trained on general image data and then applied to almond segmentation. Despite the success of this approach for almond segmentation, it had limited success on other crops such as sweet pepper [17]. Other examples include the use of DCNN features in place recognition. Chen *et al.* [18] and Sunderhauf *et al.* [19] used a pre-trained DCNN simply as a feature extractor to obtain robust features, but no further training on the DCNN was performed.

In computer vision, the task of fine-grained (species-level) image classification has highlighted the potential to transfer, or adapt, pre-trained networks to the particular task. This is in contrast to using a pre-trained network as a feature extractor such as for place recognition, as all of the parameters of the network are adapted for the task at hand. Such an approach has been at the heart of state-of-the-art fine-grained classification techniques such as joint parts localisation and image classification [20] and the recently proposed mixture of DCNNs [6], as well as state-of-the-art image segmentation approaches such as the fully connected network (FCN) of Long *et al.* [21]. Long *et al.* demonstrated that classification networks, such as AlexNet, VGG and GoogLeNet, can be treated as a 1×1 convolution. A deconvolution layer can then be integrated to upsample the classification layer (1×1 convolution) to achieve dense (per-pixel) image segmentation results. These approaches have all achieved impressive results, however, they still rely on the use of complicated network structures such as AlexNet, VGG or GoogLeNet.

An issue with the above methods is that they make use of complicated DCNNs. For instance, the relatively compact Inception-v3 model [5], which achieves impressive object classification performance (for 1000 classes), consists of 25M parameters. Other state-of-the-art DCNNs such as AlexNet and VGG are even more complicated consisting of 60M and 180M parameters respectively. In the following section we describe work towards compressing these complicated networks.

B. Model Compression

Model compression for DCNNs was first introduced in 2006 by Buclia *et al.* [22] and extended in 2014 by Hinton *et al.* [23] using the concept of model "distillation."

Model compression takes a well trained complicated model that is then used to "teach" a less complicated model, the *student*. This is achieved by changing the classification loss to an $L2$ regression loss so that the student network learns to replicate the output, $\mathbf{y}$, of the teacher given the same input, $\mathbf{X}$. The output of the teacher, $\mathbf{y}$, is its logit output, prior to the softmax for classification, and in this case the input $\mathbf{X}$ is the colour image. Minor improvements were obtained by averaging this $L2$ regression loss with the traditional classification loss.

Other work has used model compression to examine the question "do deep networks really need to be deep?", or if they can be converted into shallow networks (with only one or two layers) [24], [25]. Recent work by Urban *et al.* [25] indicates that deep network structures are still important, even for compressed models. All of the previous work using model compression has learnt the complicated *teacher* on the same dataset that the *student* is then applied to.

Alternative methods for reducing the memory size of DCNNs include the recent work of Anwar *et al.* [26] and Han *et al.* [27]. Anwar *et al.* proposed to reduce the memory of DCNNs by first pruning a network and then retraining the network to recover for the loss in performance. Han *et al.* proposed a three-stage process to reduce the memory (size) of a DCNN by applying pruning, trained quantization and finally Huffman coding. This led to considerable reductions in memory size, with the overall model size being reduced by up to 49 times without any reduction in model accuracy. Despite this impressive reduction in memory size it led to only modest speed improvements of 3.5 times.

III. Proposed Methods

In this paper we propose a novel approach for training deep convolutional neural networks (DCNNs) that allows us to trade-off both speed and memory size against accuracy. This is achieved through a three-step process.

1) We adapt a pre-trained model to the task at hand. In this work we adapt the Inception-v3 [5] model and refer to this adapted model as Adapted-IV3. The process for adaptation is described in Section III-A.
2) The adapted model, Adapted-IV3, is then used to *teach* (train) a much smaller (lightweight) DCNN. The way in which the adapted network is used to teach the smaller network is described in Section III-B.
3) Inspired by [6], we combine a set of K lightweight models as a mixture model to further enhance the performance of the lightweight models. This involves further training which is described in Section III-C.

We apply our approach to the problem of weed segmentation for robotic platforms such as AgBot II. Based on prior art [3], we extract regions based on a sliding window[1] of size

[1]Haug *et al.* used a window of $W = 80$, we use $W = 81$ to ensure a symmetric window around the location of interest.

$W \times W \times 3$ across the colour image and the class of the central pixel is declared (either weed or crop). The FCN approach is not trialled here as it is designed to be trained and evaluated on problems where dense (for every pixel) decisions are required; the training data requires a label for each pixel. However, many robotic vision problems are sparse problems where classification decisions are only required for some of the pixels. For example, weed segmentation only requires a classification for pixels containing vegetation, see Fig. 2.

A. Transfer Learning: Adapting Complicated Pre-Trained Networks

Adapting pre-trained networks is a common approach for fine-grained classification [6]. In [28] it was shown that when there is limited data to train a network from scratch it is better practice to take a pre-trained network, trained on a large dataset such as ImageNet (1M images for 1,000 classes), and adapt it for the task at hand. For this reason we first adapt a pre-trained network to the task of weed segmentation.

We take the latest version of GoogLeNet, Inception-v3 [5], which consists of 25M parameters. This model was initially trained on the ImageNet dataset using 1000 classes each with 1000 images and achieved impressive classification performance. The Inception-v3 model was derived for mobile applications and so consists of a low number of parameters, however, it is still quite a complicated model. To reduce the number of parameters and improve the speed of the network while retaining high accuracy we apply model compression, where the adapted Inception-v3 model acts as the *teacher*.

When performing transfer learning we have to upsample the original images to match the template required by the Inception-v3 architecture which is $W_{IV3} = 299$ (a $299 \times 299 \times 3$ image); this high resolution input is referred to as $\mathbf{X}_{IV3}$. To do this we upsample the much smaller image size of $W = 81$ referred to as $\mathbf{X}_{orig.}$ (an $81 \times 81 \times 3$ image).

B. Model Compression: Training Lightweight DCNNS

Model compression is often performed by first training a complicated classifier for a dataset and then training a less complicated student network on the same, or a subset, of this dataset [22], [23] (see Section II-B). In this work, we propose to instead adapt a complicated pre-trained model to the particular task at hand, this is the *teacher* network. We then learn a lightweight *student* DCNN from this model.

The *teacher* network takes as input $\mathbf{X}_{IV3}$ and produces the logit output $\mathbf{y}_{IV3}$, prior to applying the softmax operation. The *student* network takes as input $\mathbf{X}_{orig.}$ and produces the logit output $\mathbf{y}^*$, prior to applying the softmax operation. For training, both the classification loss (between the *student* and groundtruth) and the $L2$ loss (between $\mathbf{y}_{IV3}$ and $\mathbf{y}^*$) are used. This allows the *student* to represent the input signal in the same manner as the *teacher* while still requiring the classification accuracy to be optimised.

The *student* network uses the original image size, $\mathbf{X}_{orig.}$, as the upsampling operation to obtain $\mathbf{X}_{IV3}$ was only performed to match the required input size of the Inception-v3 network. This means that the *student* network is already less complicated as it is operating on a lower resolution image.

The *student* network is a lightweight DCNN that is still a deep network and we consider two potential structures. The structures for these two networks have been chosen so that the advantages of a DCNN, increasingly complex representation with a low number of parameters, are still retained. Each layer is kept to a low number of parameters as the DCNN is designed to solve one particular task (weed segmentation) rather than general object classification of 1000 classes.

The first network structure consists of 8 convolutional layers of increasing complexity (number of parameters) and 1 fully connected layer, similar to AlexNet. An illustration of this network is given in Fig. 3 (top). We refer to this as AgNet as it has been derived with agricultural tasks in mind.

The second network structure consists of 4 convolutional layers followed by 2 inception-style modules (similar to GoogLeNet) and 1 fully connected layer. An illustration of this network is given in Fig. 3 (bottom). We refer to this a MiniInception as it incorporates two inception modules. The inception modules are inspired by [5][2]. This consists of 4 sub-modules, illustrated in Fig. 4, including a 5 convolution (made up of two 3 convolutions), a 3×3 convolution, a pooling layer followed by a 1×1 convolution and a 1×1 convolution. Details of the parameters used for these inception modules can be found in Table I.

C. Mixtures and Ensembles of DCNNs

To enhance the performance of the compressed models (AgNet or MiniInception) we use an Ensemble of these lightweight models. Ensembles of K DCNNs is often used to improve the performance of state-of-the-art object classifiers [5] as the randomness of the initialisation and training process will lead to each DCNN providing slightly different decisions. Taking the average of these decisions consistently improves performance at the cost of linear complexity with the number of Ensembles, K.

We propose to further enhance the benefits of combining K DCNNs by using the recently proposed Mixture of DCNNs structure [6] (MixDCNN). The MixDCNN structure combines K DCNNs by weighting the contribution of each DCNN by their confidence for predicting the t-th sample. To do this, an occupation probability α^t is calculated based on the maximum logit value from each of the K DCNNs[3],

$$\alpha_k^t = \frac{\exp\{C_k\}}{\sum_{j=1}^{K} \exp\{C_j\}} \tag{1}$$

where C_k is the maximum logit value from the k-th DCNN. The final classification is given as a mixture of the classification decisions from the K networks, weighted by their respective occupation probability,

$$c_n = \sum_{k=1}^{K} \alpha_k^t c_{k,n}, \tag{2}$$

[2]In particular the architecture illustrated in Fig. 5 of [5].

[3]The logit value is used rather than softmax as the softmax is a normalised value.

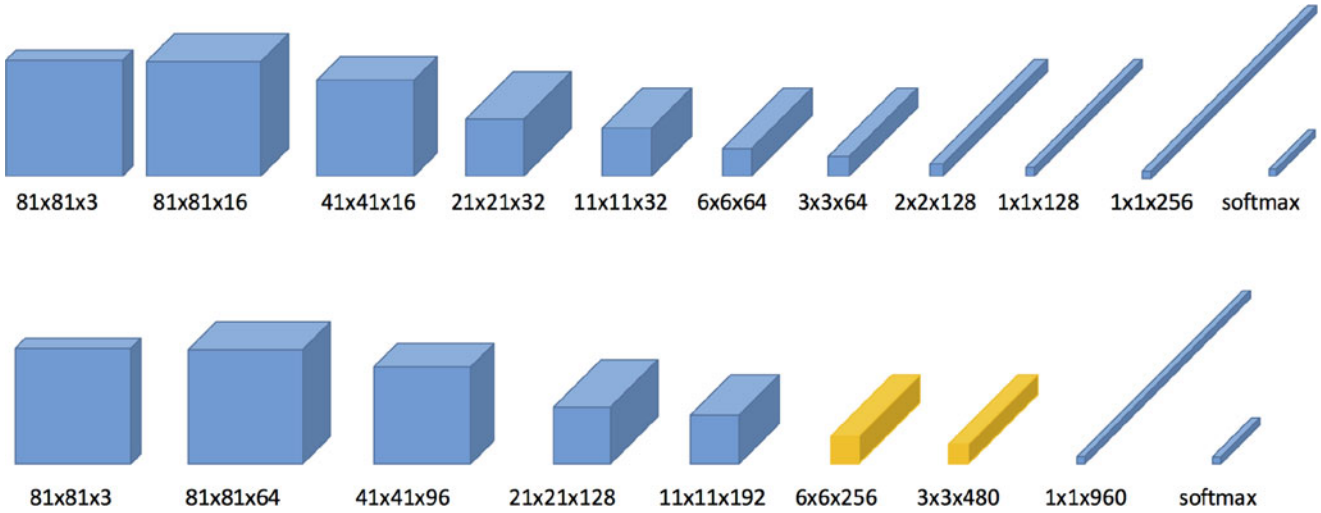

Fig. 3. An illustration of the AgNet and MiniInception DCNNs. Top is the AgNet model which is a deep model consisting of 8 convolutional layers and one fully connected layer. Despite the depth of this model it is lightweight as it has less than 0.25M parameters. Bottom is the MiniInception model which has a similar structure to the AgNet model, however, the later convolutional layers incorporate two inception modules (highlighted in yellow). This leads to a more complicated model consisting of 5.1M parameters.

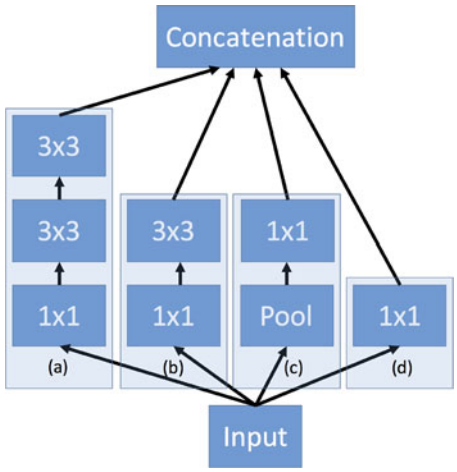

Fig. 4. An illustration of the Inception module that we use, it consists of four sub-modules (a)–(d). The kernel size for each of the sub-modules is smaller than the input, however, by concatenating them at the end the output feature vector is larger. Module (a) acts as a 5×5 convolution that incorporates downsampling of 2, it is calculated efficiently by stacking two 3×3 convolutions. Module (b) is a 3×3 convolution that downsamples by a factor of 2 and (c) first pools to downsample by a factor of 2 followed by a 1×1 convolution. Module (d) is a 1×1 convolution that dowsamples the image by a factor of 2.

TABLE I
THE NUMBER OF PARAMETERS FOR THE DIFFERENT PARTS OF THE SUB-MODULES FOR THE TWO INCEPTION MODULES USED FOR THE MINIINCEPTION DCNN

		Sub-Module			
		(a)	(b)	(c)	(d)
Inception Module 1	(1×1)	16	96	32	64
	(3×3)	32	128		
	(3×3)	32			
Inception Module 2	(1×1)	32	128	64	128
	(3×3)	96	192		
	(3×3)	96			

For Sub-module (a) the Number of Parameters are (from Top to Bottom) for the $1times1$ Layer, and then for the two 3×3 Layers. For the Sub-module (b) the Number of Parameters are for the $1times1$ Layer, and then the 3×3 Layer. For Sub-modules (c) and (d) the Number of Parameters are for the 1×1 Layers.

where $c_{k,n}$ is the probability of the sample belonging to the n-th class for the k-th component and c_n is the final classification probability for the n-th class. Using this MixDCNN structure, we combine and re-train the K DCNNs.

IV. EXPERIMENTAL RESULTS

We apply our approach to the problem of weed segmentation for robotic platforms such as AgBot II and produce results on the publicly available weed/crop dataset [7]. All of our models were implemented in Tensorflow[4]. The AdamOptimizer [29] was used with a learning rate of $\gamma = 1e^{-4}$, $\epsilon = 0.1$, a batch size of $b = 60$ and a dropout rate of 50%. When training the lightweight DCNNs we use the same data as was used to fine-tune the complicated (Inception-v3) network.

A. Accuracy of Weed Segmentation

Weed segmentation is key to enabling integrated weed management for intra-row weed management. This task is made challenging by the fact that the weeds can overlap with the crop making accurate classification difficult, see Fig. 2. To explore the performance of weed segmentation we make use of the publicly available Crop/Weed Field Image Dataset (CWFID) [7] which consists of 20 training images and 40 testing images; we make use of the agriculture protocol. We compare to the baseline method presented by Haug and Ostermann [7] which trains a random forest classifier on features obtained from shape and pixel intensity information. The number of convolutional parameters for each layer of the lightweight AgNet model

[4]TensorFlow r0.10: https://www.tensorflow.org/ using a GeForce Titan X graphics card.

TABLE II
CLASSIFICATION ACCURACY FOR THE VARIOUS MODELS ON THE CROP/WEED FIELD IMAGE DATASET (CWFID) [7]

	Accuracy	Num. Params
Adapted-IV3	**93.9**%	25M
Mix-MiniInception ($K = 4$)	90.7%	20.4M
Mix-MiniInception ($K = 2$)	90.5%	10.2M
MiniInception	89.9% $\pm$ 0.1%	5.1M
Mix-Agnet ($K = 8$)	90.5%	2.0M
Mix-Agnet ($K = 4$)	90.3%	1.0M
Mix-Agnet ($K = 2$)	89.7%	0.5M
AgNet	88.9% $\pm$ 0.4%	0.25M
Shape+Stat. [7]	85.9%	N/A

Also presented is the model complexity in terms of number of parameters (Num. Params). The accuracy for AgNet is presented as the mean accuracy and standard deviation of the 8 trained AgNet models.

were $[16, 16, 32, 32, 64, 64, 128, 128]$ with the fully connected layer consisting of 256 neurons. The parameters for the MiniInceptions were $[64, 96, 128, 192]$ for the convolutional layers, $[240, 480]$ for the two inception modules (see Table I and Fig. 4 for more details) followed by a fully connected layer consisting of 960 neurons.

The results in Table II highlight the potential improvements available by using a deep neural network based solution. It shows that all of the presented deep learning solutions considerably outperform the previously proposed method, with the highest accuracy achieved by the Adapted-IV3 model with 93.9% compared to 85.9% when using the previous approach proposed by Haug and Ostermann [7].

The downside of the Adapted-IV3 system is that it is a complex model with approximately 25M parameters. By contrast, a lightweight model can be trained using the Adapted-IV3 model as a *teacher* to achieve an average accuracy of 88.9% $\pm$ 0.4% for the AgNet structure (0.25M parameters) and 89.9% $\pm$ 0.1% for the MinInception structure (5.1M parameters); we present the mean accuracy and standard deviation of the 8 trained AgNet models and 4 trained MiniInception models. Although these lightweight models have lower performance they also consist of 100 and 5 times fewer parameters for the lightweight AgNet and MiniInception models respectively. This makes these models more appropriate for a robotic and mobile solution, such as AgBot II. This is because reducing the number of parameters reduces the memory requirements making it a candidate for less powerful graphical processing units (GPUs) and second by lowering the number of parameters we also decrease the execution time. We will elucidate on this second point, the decrease in execution time, in Section IV-C.

B. Mixtures of Lightweight DCNNs

To enhance the performance of the compressed models (AgNet and MiniInception) we combine multiple lightweight models as a mixture model, referred to as Mix-AgNet and Mix-MiniInception. Similar to [6], our empirical evaluations found that the MixDCNN approach always outperforms a simple Ensemble. For our experiments, the MixDCNN approach provided an absolute average improvement in accuracy of 0.15% compared to using an Ensemble; when deploying on a single GPU there is no added model complexity between the Ensemble and MixDCNN approach.

It can be seen in Table II that increasing K, the number of models, increases the overall performance of the system. However, the relative performance improvement decreases as K increases. A good tradeoff between model complexity and accuracy is to use $K = 4$ AgNet DCNNs as this improves the performance to 90.3% but results in just 1.0M parameters, this is 25.0 times fewer than the Adapted-IV3 model. For the MiniInception model using $K = 2$ DCNNs improves the performance to 90.5% but results in 10.2M parameters, this is 2.5 times smaller than the Adapted-IV3 model. Doubling to $K = 8$ and $K = 4$ for the AgNet and MiniInception models respectively results in an absolute improvement in accuracy of just 0.2% but at the cost of doubling the number of parameters. If the choice of model is based purely on the number of parameters then the AgNet and Mix-AgNet models provide the best tradeoff between the number of parameters and accuracy, however, as we will show in the next section if speed is the criteria the tradeoff is more complicated.

A visualisation of the results from the three DCNN models, AgNet, MixAgNet ($K = 4$), and Adapted-IV3, are provided in Fig. 5; more visualisation results are available in the video.

C. Speed and Model Complexity

In Table III the number of regions that can be processed per second (reg./sec.) is presented. It can be seen that the less complicated models are able to process more samples, for instance AgNet which consists of 0.25M parameters can process 6300 regions (pixels) per second, this is approximately 40.6 times more than the Adapted-IV3 model. By contrast, the MiniInception model which consists of 5.1M parameters can process 2737 regions (pixels) per second, this is approximately 17.7 times more than the Adapted-IV3 model. This shows that there is not always a linear relationship between the speed of a model and the number of parameters (complexity) as the MiniInception model has 20.4 times the number of parameters but is only 2.3 times slower.

Increasing the complexity, by adding more models, decreases the number of regions that can be processed per second. This provides us with a method to tradeoff accuracy and run-time speed. For the highly accurate Adapted-IV3 model we can achieve an accuracy of 93.9%, however, we are only able to process 155 regions per second. By contrast, the Mix-AgNet $K = 4$ model can achieve an accuracy of 90.3% but is able to process 15.8 times regions per second (2445 vs 155) and the Mix-MiniInception $K = 2$ model can achieve an accuracy of 90.5% while processing 9.3 times the regions per second (1437 vs 155).

In terms of frames per second, we first note that when performing weed segmentation often only a subset of the potential vegetation regions are classified. In [3] the image was subsampled using a sparse grid of 15×15. Considering such an approach but instead using a stride of $S = 8$ (horizontally and vertically) between region evaluations, the average number of

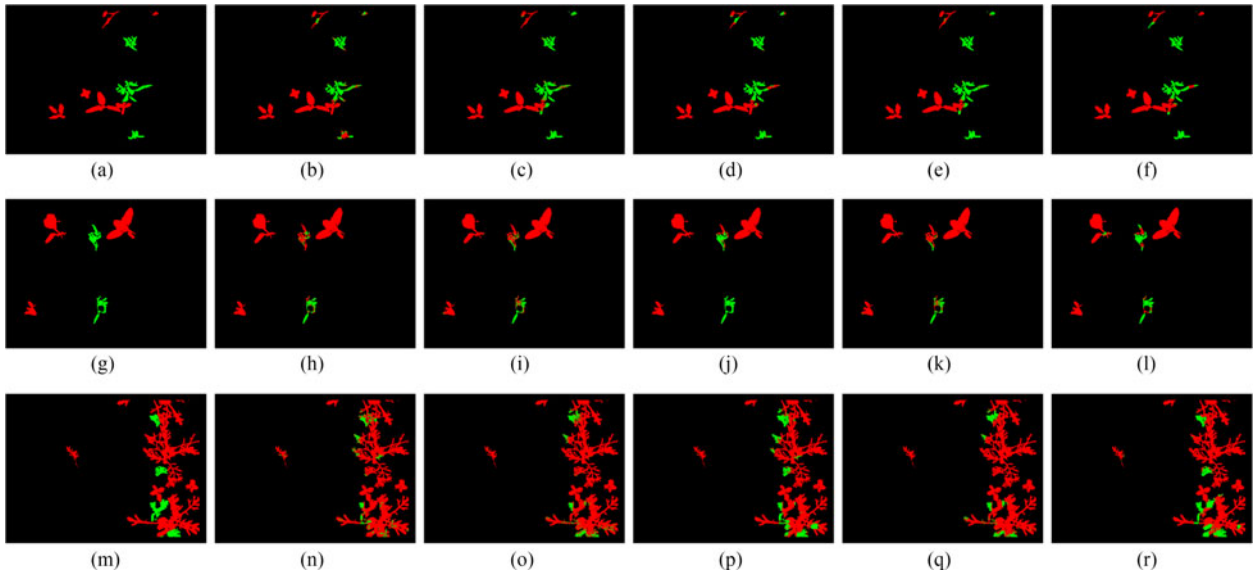

Fig. 5. Example output of the three different deep learning approaches, red represents the weeds and green represents the crop. From left to right is results for the ground truth, AgNet, MiniInception, Mix-AgNet ($K = 4$), Mix-MiniInception ($K = 2$) and the Adapted-IV3 models. Each row represents the result for a different image, where the last row represents a failure case for the lightweight models.

TABLE III
NUMBER OF PARAMETERS (NUM. PARAMETER) AND TIMING IN TERMS OF REGIONS PROCESSED PER SECOND (REG./SEC.)

	Num. Parameters	reg./sec.
Adapted-IV3	25M	155
Mix-MiniInception ($K = 4$)	20.4M	767
Mix-MiniInception ($K = 2$)	10.2M	1,437
MiniInception	5.1M	2,737
Mix-AgNet ($K = 8$)	2.0M	1,394
Mix-AgNet ($K = 4$)	1.0M	2,445
Mix-AgNet ($K = 2$)	0.5M	3,736
AgNet	0.25M	6,300

regions to be classified per image for the CWFID dataset is 1,337. This allows our proposed approaches to operate at the following frames per second (fps): 4.71 for AgNet, 1.83 for Mix-AgNet $K = 4$, 1.07 for Mix-MiniInception $K = 2$ and 0.12 for Adapted-IV3.

From this it can be seen that the Adapted-IV3 approach, which operates at just 0.12 fps, is usable either as a post-processing method or for very slow moving vehicles. By comparison, the AgNet, Mix-AgNet and Mix-MiniInception approaches have the potential to be deployed on robotic platform such as AgBot II as they can operate between 4.73 and 1.07 fps.

V. CONCLUSION

We have proposed a novel approach for training DCNNs that allows us to trade-off memory size and speed against accuracy by learning a set of lightweight models. The key approach for training these models is to "distill" the knowledge from a complex pre-trained model considered to be an expert for the task at hand. This *teacher* is the used to train a much smaller (lightweight) *student* DCNN, AgNet. A set of K-lightweight models can then be combined as a mixture model, MixAgNet, to further enhance the performance of the lightweight models.

Using this approach we have been able to considerably improve the accuracy of robotic weed segmentation from 85.9%, using a traditional approach, to 93.9% using an adapted DCNN (Adapted-IV3) with 25M parameters able to process 155 regions per second. The downside to this highly accurate model is that it is current not able to be deployed for real-time processing on a robotic system (such as AgBot II) as it can only process 0.12 frames per second. However, we have shown that this complex model can be made scalable (in terms of memory and speed) by learning $K = 4$ lightweight AgNet models or $K = 2$ Mini-Inception models leading to an accuracy of 90.3% and 90.5% while using 25.0 and 2.5 times fewer parameters respectively. These smaller, yet still accurate, models are able to process 1.83 and 1.07 frames per second respectively.

Future work will examine how to improve the accuracy of the lightweight DCNNs. Alternative formulations of the lightweight DCNN will be examined including if greater depth needs to be introduced into the network as this was shown to be important [25]. Finally, the applicability of this approach to derive efficient versions of an FCN structure could also be explored; at its core the FCN consists of a complicated DCNN such as VGG or GoogLeNet.

REFERENCES

[1] O. Bawden, "Design of a lightweight, modular robotic vehicle for the sustainable intensification of broadacre agriculture," Master's thesis, Queensland Univ. Technol., St Lucia, QLD, Canada, 2015.
[2] C. Lehnert, A. English, C. McCool, A. Tow, and T. Perez, "Autonomous sweet pepper harvesting for protected cropping systems," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 872–879, Apr. 2017.
[3] S. Haug, A. Michaels, P. Biber, and J. Ostermann, "Plant classification system for crop/weed discrimination without segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2014, pp. 1142–1149.
[4] Y. LeCunn, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[5] C. Szegedy, V. Vanhoucke, S. Ioffe, and S. Shlens, Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, 2015, pp. 2818–2826.
[6] Z. Ge, A. Bewley, C. McCool, P. Corke, B. Upcroft, and C. Sanderson, "Fine-grained classification via mixture of deep convolutional neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2016, pp. 1–6.
[7] S. Haug and J. Ostermann, "A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks," in *Proc. ECCV Workshop Comput. Vis. Probl. Plant Phenotyping*, 2014, pp 105–116.
[8] S. A. Shearer, "Plant identification using color co-occurrence matrices derived from digitized images," Ph.D. dissertation, Ohio State Univ., Columbus, OH, USA, 1986.
[9] R. Zwiggelaar, "A review of spectral properties of plants and their potential use for crop/weed discrimination in row-crops," *Crop Protection*, vol. 17, pp. 189–206, 1998.
[10] C. Lin, "A support vector machine embedded weed identification system," Ph.D. dissertation, , Univ. Illinois, IL, USA, 2009.
[11] F. D. Rainville *et al.*, "Bayesian classification and unsupervised learning for isolating weeds in row crops," *Pattern Anal. Appl.*, vol. 17, pp. 401–414, 2014.
[12] D. Hall, C. McCool, F. Dayoub, N. Sunderhauf, and B. Upcroft, "Evaluation of features for leaf classification in challenging conditions," in *Proc. IEEE Int. Winter Conf. Appl. Comput. Vis.*, 2015, pp. 797–804.
[13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2001, pp. I-511–I-518.
[14] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learning Theory*, 1995, pp. 23–37.
[15] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.
[16] C. Hung, J. Nieto, Z. Taylor, J. Underwood, and S. Sukkarieh, "Orchard fruit segmentation using multi-spectral feature learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 5314–5320.
[17] C. McCool, I. Sa, F. Dayoub, C. Lehnert, T. Perez, and B. Upcroft, "Visual detection of occluded crop: For automated harvesting," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 2506–2512.
[18] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional neural network-based place recognition," in *Proc. Australas. Conf. Robot. Autom.*, 2014.
[19] N. Sunderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 4297–4304.
[20] D. Lin, X. Shen, C. Lu, and J. Jia, "Deep LAC: Deep localization, alignment and classification for fine-grained recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1666–1674.
[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[22] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 535–541.
[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Workshop*, 2014.
[24] L. Ba and R. Caruana, "Do deep convolutional nets really need to be deep?" in *Proc. Neural Inf. Process. Syst.*, 2014.
[25] G. Urban *et al.*, "Do deep convolutional nets really need to be deep (or even convolutional)?" in *arXiv*, 2016.
[26] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," in *ACM J. Emerg. Tech. Comput. Syst.*, vol. 13, no. 3, 2017.
[27] S. Han, M. H, and W. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. ICLR*, 2016.
[28] Z. Ge, C. McCool, C. Sanderson, and P. Corke, "Subset feature learning for fine-grained category classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop*, 2015, pp. 46–52.
[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.