# Insects Image Classification through Deep Convolutional Neural Networks

Francesco Visalli[1], Teresa Bonacci[2], and N. Alberto Borghese[1]

[1] Department of Computer Science, Università degli Studi di Milano, Italy,
francesco.visalli@studenti.unimi.it, alberto.borghese@unimi.it
[2] DiBEST Department, Università della Calabria, Italy
teresa.bonacci@unical.it

**Abstract.** We present and discuss results of the application of a deep convolutional network model developed for the automatic recognition of images of insects. The network was trained using transfer learning on an architecture called MobileNet, specifically developed for mobile applications. To fine tune the model, a grid-search on hyperparameters space was carried out reaching a final accuracy of 98.39% on 11 classes. Fine-tuned models were validated using 10-fold cross validation and the best model was integrated into an Android application for practical use. We propose solving the "open set" problem through feed-back collected with the application itself. This work also led to the creation of a well-structured image dataset of some important species/genera of insects.

**Keywords:** Visual recognition, insect classification, deep learning, convolutional neural networks

## 1 Introduction

Insects are the largest class of the animal kingdom in our planet counting over a million of species. Within this vast taxonomic group of animals we find many vectors of various pathogens responsible for diseases and zoonosis; some of these insects infest foodstuffs and stored products, others cause the loss of entire crops and alter the quality of cultivated products and their derivatives. In order to limit their impact, a timely intervention is often essential. For this reason, tools that would allow the immediate classification of specific species of insects are needed.

This problem can be recasted as an image classification task, that is an instance of supervised learning problems. In classification tasks, we train a model $f : X \rightarrow Y$ on a set of labeled data, called training set, composed of pairs $(x, y)$ where $y \in Y$ is the correct label for the instance $x \in X$. A supervised learning algorithm modifies progressively $f(.)$ to minimize an error, associated to misclassification and specific for the task so that the correct label $y \in Y$ can be assigned to any input instance of $x \in X$.

Convolutional Neural Networks (CNNs) are kinds of multi-layer neural network architectures specialized in finding patterns within images. They have become the state of the art for image classification since 2012, when AlexNet [12] won the ImageNet Large Scale Visual Recognition Competition[3] (ILSVRC) beating others no neural network methods and achieving an error rate of 16.4% on top-5 [16].

Since then, the enormous strand of learning algorithms working on networks endowed with many layers (named deep networks) has begun for the solution of visual recognition tasks. These models obtain remarkable results thanks also to the increase in GPUs computing power that allows the execution of complex learning algorithms in reasonable amount of time.

We propose here a particular CNN fine-tuned on a MobileNet [8] architecture, that was itself pre-trained on ImageNet. MobileNet is an architecture designed specifically for mobile applications. This architecture was chosen for its simplicity, because it can run on smart phones and for future experiments.

Because there are no specific databases of insects, we built our own dataset, that counts 13,588 images (belonging to one of 11 classes) and we leveraged this dataset to train MobileNet. Different hyper-parameters were used. The models obtained were validated through 10-fold cross validation and the best model was integrated into an Android application.

Thanks to this application we have been able to propose an operational transversal solution to an open problem in visual recognition: the "open set" recognition [19]. This occurs when we have to recognize if an image belongs to one of the specified classes or it belongs to a different external class. This information can be collected by the users themselves who take a picture of insects from their smart-phone, see the current classification proposal of the system with the associated degree of confidence, and, in case the insect has not been correctly classified, he/she can report this to the system that can refine, in batches, the classification output.

Finally, in order to investigate the nature of the features that the network learnt, we tested our model on a small dataset of insects that do not belong to the classes identified, including various insects similar to those present in the dataset on which the network was trained.

## 2 Related Work

CNNs trained on large datasets such as ImageNet, that include a large number of images of insects, have already been proved to be capable of classifying them. However, to the best of our knowledge, there are few existing works [7, 14], specialized in the automatic classification of images of these invertebrates. Moreover, there are not satisfying works dealing with the "open set" problem, that for insects that belong to millions of species, is a real issue.

In addition, there was the need to create an accurate and well-structured dataset targeted to classification of insects of medical and agronomic interest, taking into account that at certain levels of detail even the taxonomist expert needs a thorough morphological study for the identification of species.

[3] http://www.image-net.org/challenges/LSVRC/

## 3 Methodology

### 3.1 Transfer Learning

Training a neural network from scratch is a difficult task which requires a huge amount of resources as computational power, lot of data, time, and frequently leads to overfitting. These kinds of neural networks are also rich in hyperparameters that need to be tuned. To solve these issues we trained our network using transfer learning [15, 4, 20]. This is a machine learning method that allows the knowledge transfer between domains; in neural networks knowledge is embedded in the network weights. The idea behind transfer learning on CNNs is that features learned in the first layers are often the same regardless of the domain. The more similar are domains, the more are the shared features, the less is the training needed.

We exploited two aspects of transfer learning: first we trained, on the top of a MobileNet [8], a Softmax classifier. Such classifier allows interpreting its output as a degree of similarity of the image with one of the identified 11 classes. Then, we fine-tuned MobileNet specializing it on the features of insects of our interest.

### 3.2 MobileNet

We chose the MobileNet architecture because it was designed for mobile development. In particular, it can do the forward pass on client side. Furthermore, its simplicity makes it a good candidate for future experiments and developments.

MobileNet is a 28 layers CNN that uses depth-wise separable convolutional layers instead of classical convolutional layers. Depth-wise separable convolution operation splits the convolution operation in two phases: first, the input is filtered in a depth-wise convolutional layer, then the output of the first phase is combined in a separate point-wise convolutional layer. In classical convolution these two steps are merged into one. This split reduces the number of operations required for the convolution without degradates performances. This is particularly suitable to RGB images where convolution can be computed separately on the three different channels. For more details on depth-wise separable convolution we refer to the original paper [8].

The first layer of the net is a classic convolutional layer, then there are 26 layers alterning depth-wise convolution and point-wise convolution. Between a convolution operation and another one, there is a batch normalization operation followed by a ReLU6 activation function The ReLU 6 is a variant of the classical ReLU which makes the network more robust than regular ReLU when using low-precision computation [11].

The last layer of the network is a fully connected layer preceded by a global average pooling operation. The final classification is made by a Softmax classifier.

## 4 Training

### 4.1 Dataset

There are a lot of challenges that any algorithm of image classification has to face off including: image occlusion, illumination conditions, image deformation, viewpoint variation, scale variation, intra-class variation, background clutter and so on. In supervised learning we try to address these issues through a data-driven approach. To correctly address such problems, the dataset on which the algorithm learns should be as various as possible and cover all possible variants.

Our dataset is composed of 11 classes. Each class counts a mean of 1000 images (except one that was not present in the laboratory when the dataset was created) for a total amount of 13,588 images. We collected all images for insects of our interest from Google Images through a script, and then we cleaned the results by hand. We integrated these images with photos took in the laboratory (Figure 1). The classes included into the dataset are the following: *Brachinus* sp., *Chrysolina* sp., *Chrysomela* sp., *Cucujus* sp., *Graphosoma* sp., *Cucujus* sp., *Leptinotarsa decemlineata*, *Nezara viridula*, *Pyrochroa* sp., *Rhynchophorus ferrugineus*, *Vespa crabro*, *Vespula* sp. The classes were chosen because they are dangerous to agriculture, dangerous to other animal and plant species or simply because they are common.

The dataset was splitted, in a random but reproducible way[4], into training set and test set using 90% and 10% of images for each class, respectively.

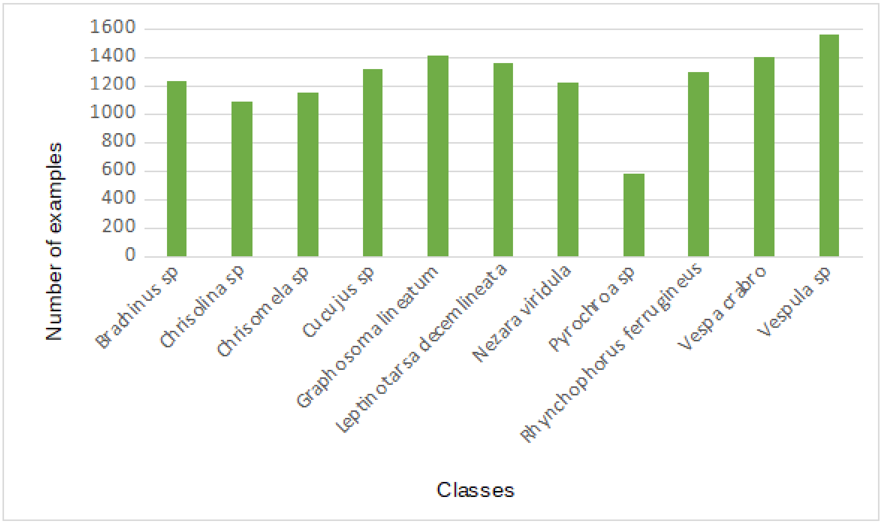


**Fig. 1.** Number of examples within the classes.

[4] https://cs230.stanford.edu/blog/split/

### 4.2 Training Settings

In order to train our network we leveraged Keras[5] with TensorFlow[6] as back-end. Keras is an open-source, high level library, written in Python, for deep learning. It provides several models. We focused on CNN models pre-trained on ImageNet. In particular, we leveraged the MobileNet architecture described above.

In order to achieve the best accuracy, we set the multiplier width and resolution to 1. We used the cross entropy function as loss function that is the best choice using Softmax classifiers.

The dataset was constructed in such a way that the number of examples within the classes is balanced (Figure 1). Moreover, given the nature of the problem, we used the accuracy as metric to evaluate the results.

Every training run ends with early stopping, 10% of training set images for each class was used to build the validation set. We monitored the loss on the validation set and stopped the training process after 4 epochs that the loss did not decrease anymore.

Models were trained and validated on a NVIDIA GeForce GTX 950M.

### 4.3 MobileNet as Features Extractor

The network was trained using transfer learning method. As first step of our training, we exploited the MobileNet CNN pre-trained on ImageNet as features extractor. We replaced the last layer of MobileNet provided by Keras with our 11 outputs Softmax classifier. As previously mentioned, in ImageNet there are a lot of images of insects, so we can reasonably suppose that the already learned features could guarantee good results already at this step.

We froze all the weights of the network and did a grid-search for tuning hyper-parameters in the following space:

$$\text{learning rate: } \{log_{10}b = c\}, c \in \{-6, ..., -1\}$$
$$\text{batch size: } \{2^a\}, a \in \{3, ..., 6\}$$
$$\text{optimizers: } \{Adam, RMSProp\}$$

Because the learning rate is a multiplier, it is usually searched in logarithmic space, while the batch size is commonly set as power of 2 for efficient computation. We adopted two optimizers here: Adam algorithm [10] is one of the best choice to optimize the gradient descent in CNNs and RMSprop [3] is the optimizer used in the original MobileNet paper. The hyper-parameters of the optimizers are the default ones suggested by Keras.

Let's introduce the tuple $(optimizer, learningrate, batchsize)$ to define a configuration of hyperparameters. Table 4.3 shows results about the two best models calculated on the training and validation set. The best hyper-parameters configurations are: $(Adam, 10^{-4}, 2^3)$ and $(RMSProp, 10^{-4}, 2^4)$. Training results are interesting already in this phase. The best results are obtained with batch sizes

[5] https://keras.io/
[6] https://www.tensorflow.org/

in $\{2^3, 2^4\}$ and with learning rate in $\{10^{-5}, 10^{-4}, 10^{-3}\}$. Learning rate of $10^{-1}$ was too high, whereas a learning rate of $10^{-6}$ was too slow in convergence. The grid-search took about 3 days.

**Table 1.** Training results of the Softmax classifier on training and validation set.

| Configuration | Epochs | Training Set | | Validation Set | |
|---|---|---|---|---|---|
| | | Loss | Accuracy | Loss | Accuracy |
| $(Adam, 10^{-4}, 2^3)$ | 5 | 0.0599 | 0.9842 | 0.0908 | 0.9737 |
| $(RMSProp, 10^{-4}, 2^4)$ | 10 | 0.0081 | 0.9982 | 0.1066 | 0.9742 |

### 4.4 MobileNet: Fine Tuning

In this phase we fine-tuned the best models obtained in the previous step leaving the weights of the network free to vary. In such way, the network can learn features for the classification of the insects specific of this application. Since we have a fairly large dataset we fine-tuned all the layers of MobileNet.

Results of the previous phase suggested us to use a small batch size and a small learning rate. We searched hyper-parameters through grid-search in the following space:

$$\text{learning rate: } \{log_{10}b = c\}, c \in \{-6, ..., -3\}$$
$$\text{batch size: } \{2^a\}, a \in \{3, 4, 5\}$$
$$\text{optimizers: } \{Adam, RMSProp\}$$

The best configuration are $(Adam, 10^{-5}, 2^3)$ and $(RMSProp, 10^{-4}, 2^3)$ both of them obtained from the fine tuning of $(Adam, 10^{-4}, 2^3)$. Results about the two best models calculated on the training and validation set are presented in Table 4.4. The grid-search on both models took about three and a half days.

**Table 2.** Results of the fine tuning on training and validation set.

| Configuration | Epochs | Training Set | | Validation Set | |
|---|---|---|---|---|---|
| | | Loss | Accuracy | Loss | Accuracy |
| $(Adam, 10^{-5}, 2^3)$ | 13 | 0.0041 | 0.9995 | 0.0448 | 0.9845 |
| $(RMSProp, 10^{-4}, 2^3)$ | 7 | 5.2790e-04 | 0.9997 | 0.0460 | 0.9860 |

## 5 Validation

We validated the two models obtained from fine tuning phase through 10-fold cross validation. 10 blocks of images were randomly extracted from the initial training set. The blocks are composed of 1/10 of the total images for each class. These blocks were used in turn as a validation set using each time the 9/10 of the remaining images as training sets.

The cross validation error of $(Adam, 10^{-5}, 2^3)$, obtained by averaging the single validation loss for each run, is 0.0093, whereas the cross validation error of $(RMSProp, 10^{-4}, 2^3)$ is 0.0177. Therefore, the best model is obtained by fine tuning $(Adam, 10^{-4}, 2^3)$ (i.e. the model obtained by training only the Softmax classifier from the first phase) with hyper-parameters $(Adam, 10^{-5}, 2^3)$. The cross-validation on both models took about one day.

## 6 Results and Discussion

The model proposed here is based on stacking convolutional layers one on top of the other, adding eventually a pooling layer in between. Although convolutional neural networks have been popularized inside the deep-learning domain, they were proposed in the early nineties by the group of Slotine [18] in the domain of radial basis function networks and further developed inside a hierarchical framework with real-time learning by the group of Borghese [5, 6, 2]. Similar concepts are also well-known in the mathematical domain where functional approximation through function bases is largely adopted.

The accuracy calculated on the test set with the best model is 98.39%. This should not come as a surprise given the large number of parameters implemented by such networks. The particular nature of the task and the fact of having built a custom dataset does not allow us to compare results with any existing work.

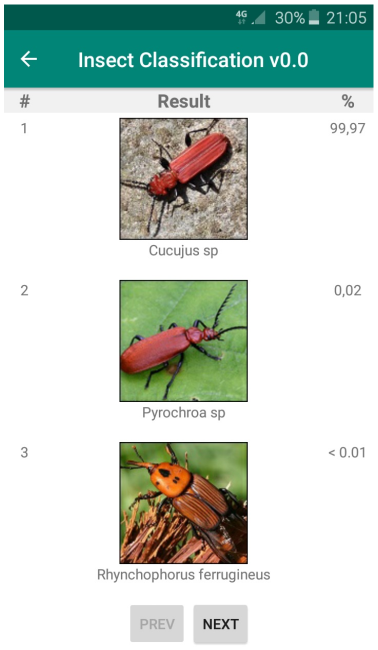
We know that as the number of classes increases, the accuracy of the network may deteriorate. However, there are many solutions that we could adopt. We could change the network architecture: MobileNet belongs to the first generation of "mobile CNNs" as well as ShuffleNet [22]. We could leverage more sophisticated networks like MobileNet V2 [17] or ShuffleNet V2 [13]. We could use the "Squeeze-and-Excitation" (SE) blocks in our MobileNet, introduced in SENet [9] that is the winner of the task of image classification of ILSVRC 2017.

In all cases, the claim would be that the numerosity of the dataset should be increased to improve the recognition rate. This is a mantra of all deep-learning algorithms and it resembles the mantra of classical Artificial Intelligence, for which any artificial intelligence would approach human intelligence provided that as complex enough local function is implemented inside it. We remark that some peculiar characteristics of human brain and reasoning are missing in these pictures.

### 6.1 Application

The classification network was integrated into an Android application (Figure 2) to field test the model and for practical use. It allows loading a picture from the phone's memory or take a picture on the spot. Once the image is chosen, in order to get a better classification, the application allows selecting the portion that contains the insect.

Results of the classification are presented displaying the thumbnail prototypical image of each class and the probability value associated. This would tackle also the problem of "open set" recognition: what would happen if we try to classify an insect that is not part of the dataset? The model, due to the nature of the classification task, would give a result based on the most similar matching between the image and the features that the network learnt.



| # | Result | % |
|---|---|---|
| 1 | Cucujus sp | 99,97 |
| 2 | Pyrochroa sp | 0,02 |
| 3 | Rhynchophorus ferrugineus | < 0.01 |

**Fig. 2.** Screenshot of the application, example of classification.

There are two possible cases: the insect is not into the dataset or the image is misclassified. In both cases, we will provide a feedback mechanism with which the user can send a report containing the image and a top-5/top-10 of the results. In this way if the insect is not into the dataset, we could decide to insert it. Otherwise, if the algorithm misclassifies the image, we will have the opportunity to investigate why the image was not correctly classified and to integrate the dataset with different images for a future training. Therefore, we can say that the thumbnail provides a self-evaluation mechanism for the user and allows increasing the size of the database with annotated images easily.

## 6.2 Test on the features

Neural networks, particularly CNNs, are a sort of black box. While first layer features are human readable, those of deeper levels are hard to understand. Some efforts have been done to try to understand how CNNs work internally, and it is catalogue under "open deep-neural network" research stream [21]. However, no results are provided up to now on internal codes used by such networks. This shortcoming is shared with classical neural networks for which the output of hidden layers was almost never explored with a few remarkable exception (e.g. [23]). We remark here that, given the large number of parameters in the network, the same results can be obtained with different outputs of the hidden layers [1] and it is not clear yet if there is any hidden output that is biologically plausible, contains a certain code and it is common across different peoples brain, or hidden output can vary largely from individual to individual.

We classified 19 external classes of images that were similar to those into the training set. Results were predictable: the network looks for shape/color matchings between taining and external classes (Figure 3). The external classes are

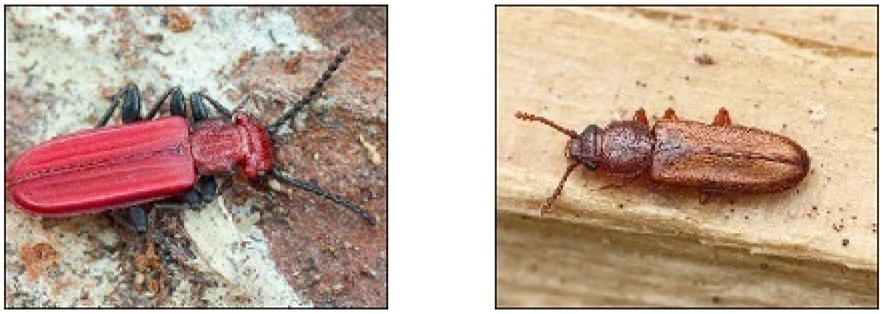


**Fig. 3.** On the left a training class (*Cucujus* sp.), on the right an external class (*Pediacus depressus*) mostly classified as *Cucujus* sp. because of its shape.

the following: *Adalia bipunctata*, *Aelia acuminate*, *Anchomenus dorsalis*, *Carpocoris pudicus*, *Corizus hyoscyami*, *Curculionidae*, *Dolichovespula* sp., *Eurygaster maura*, *Leistus* (*Pogonophorus*), *Lema daturaphila*, *Lilioceris* sp., *Nebria* (*Eunebria*) sp., *Pediacus depressus*, *Polistes* sp., *Pyrochroidae*, *Sulcopolistes* sp., *Tenthredo notha*, *Tenthredo scrophulariae*, *Vespa orientalis* (cf. Figure 3). Further investigation are needed in order to derive some insights on the features that the network learnt.

## 7 Conclusions

We built a custom dataset of images of insects on which we fine-tuned a mobile CNN: MobileNet. The best models from fine tuning phase were validated through 10-fold cross validation and the one with less cross validation error was tested on the test set obtaining 98.39% of accuracy. Finally, the network were integrated into an Android application. We have tried to give a transversal contribution to the open problem of "open set" recognition. We tried to classify a dataset composed of 19 external classes similar to those present into the training set in order to investigate the black box infrastructure of CNNs.

## References

1. Borghese, N.A., Arbib, M.A.: Generation of temporal sequences using local dynamic programming. Neural Networks **8**(1), 39–54 (1995). DOI 10.1016/0893-6080(94)00053-O. URL https://doi.org/10.1016/0893-6080(94)00053-O
2. Borghese, N.A., Ferrari, S.: Hierarchical rbf networks and local parameters estimate. Neurocomputing **19**, 259–283 (1998)
3. Dauphin, Y.N., de Vries, H., Chung, J., Bengio, Y.: Rmsprop and equilibrated adaptive learning rates for non-convex optimization. CoRR **abs/1502.04390** (2015). URL http://arxiv.org/abs/1502.04390
4. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In:

Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, *JMLR Workshop and Conference Proceedings*, vol. 32, pp. 647–655. JMLR.org (2014). URL http://proceedings.mlr.press/v32/donahue14.html

5. Ferrari, S., Bellocchio, F., Piuri, V., Borghese, N.A.: A hierarchical RBF online learning algorithm for real-time 3-d scanner. IEEE Trans. Neural Networks **21**(2), 275–285 (2010). DOI 10.1109/TNN.2009.2036438. URL https://doi.org/10.1109/TNN.2009.2036438
6. Ferrari, S., Maggioni, M., Borghese, A.: Multiscale approximation with hierarchical radial basis functions networks. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council **15**, 178–88 (2004). DOI 10.1109/TNN.2003.811355
7. Glick, J., Miller, K.: Insect classification with heirarchical deep convolutional neural networks convolutional neural networks for visual recognition ( cs 231 n ) (2016)
8. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR **abs/1704.04861** (2017). URL http://arxiv.org/abs/1704.04861
9. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pp. 7132–7141. IEEE Computer Society (2018). DOI 10.1109/CVPR.2018.00745. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Y. Bengio, Y. LeCun (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015). URL http://arxiv.org/abs/1412.6980
11. Krizhevsky, A.: Convolutional deep belief networks on cifar-10 (2010)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: P.L. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, pp. 1106–1114 (2012). URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
13. Ma, N., Zhang, X., Zheng, H., Sun, J.: Shufflenet V2: practical guidelines for efficient CNN architecture design. In: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (eds.) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV, *Lecture Notes in Computer Science*, vol. 11218, pp. 122–138. Springer (2018). DOI 10.1007/978-3-030-01264-9_8. URL https://doi.org/10.1007/978-3-030-01264-9_8
14. Martineau, M., Conte, D., Raveaux, R., Arnault, I., Munier, D., Venturini, G.: A survey on image-based insect classification. Pattern Recognition **65**, 273–284 (2017). DOI 10.1016/j.patcog.2016.12.020. URL https://doi.org/10.1016/j.patcog.2016.12.020
15. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: An astounding baseline for recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014, pp. 512–519. IEEE Computer Society (2014). DOI 10.1109/CVPRW.2014.131. URL https://doi.org/10.1109/CVPRW.2014.131

16. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015). DOI 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y
17. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pp. 4510–4520. IEEE Computer Society (2018). DOI 10.1109/CVPR.2018.00474. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html
18. Sanner, R.M., Slotine, J.E.: Gaussian networks for direct adaptive control. IEEE Trans. Neural Networks **3**(6), 837–863 (1992). DOI 10.1109/72.165588. URL https://doi.org/10.1109/72.165588
19. Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E.: Toward open set recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(7), 1757–1772 (2013). DOI 10.1109/TPAMI.2012.256. URL https://doi.org/10.1109/TPAMI.2012.256
20. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 3320–3328 (2014). URL http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks
21. Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T.J., Lipson, H.: Understanding neural networks through deep visualization. CoRR **abs/1506.06579** (2015). URL http://arxiv.org/abs/1506.06579
22. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pp. 6848–6856. IEEE Computer Society (2018). DOI 10.1109/CVPR.2018.00716. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zhang_ShuffleNet_An_Extremely_CVPR_2018_paper.html
23. Zipser, D., Andersen, R.A.: A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature **331**, 679–684 (1988)