Transfer Learning for Fine-grained Multi-class Classification of Macroinvertebrate

Lewis Bails
University of Aarhus
Department of Engineering
Aarhus, Denmark
lewis.bails@gmail.com

***Abstract*—The ongoing efforts of biologists to adequately monitor the water quality of aquatic ecosystems is heavily reliant on researchers' ability to properly identify macroinvertebrates and their subtle anthropogenic changes over time. The process of classifying these macroinvertebrates by humans is highly accurate but also time-consuming due to the fine-grained nature of the taxa. Past studies have investigated the effectiveness of machine learning models for automated classification, some achieving accuracies rivalling, and even surpassing, that of trained biologists in the field [3]. This paper explores the possibility of using transfer learning with a relatively small, but still highly effective, pre-trained convolutional neural network (CNN) architecture, MobileNet, for feature extraction. Comparative evaluation of classification accuracy is then performed using four machine learning models which use the feature vectors output by MobileNet's bottleneck layer. These models are multilayer perceptron networks (MLP), probabilistic neural networks (PNN), support vector machines (SVM) and extreme learning machines (ELM).**

I. Introduction

Due to the scarcity of fresh-water resources on the planet, their adequate care is of great importance to humans and animals alike. This necessitates close monitoring of these resources, for example, for the effects of pollution and eutrophication. Chemical assessments alone are not sufficient to determine the overall water quality of a given ecosystem. However, the subtle anthropogenic changes seen in the organisms which reside in such ecosystems can also be used as indictors of ecological health. The community constitution of benthic macroinvertebrates is highly sensitive to changes in water quality and, as such, are well-suited to this problem [3]. The human taxa classification method previously employed, however accurate, was also time-consuming and expensive. More recently, automated methods have reached the approximate accuracy of human classification [3] The best results for this problem have been largely dominated by models which make use of geometrical and intensity-based feature vectors [3] [7]. The feasibility of instead using a small but fast - and possibly mobile - pre-trained CNN to generate learned feature vectors, replacing an image's direct geometrical and intensity-based features, is addressed in the rest of this publication. In order to avoid overfitting on the relatively small original dataset, data enrichment is performed via image augmentation both during model training and outside of training. To further improve the quality of feature vector representations, fine-tuning of MobileNet's top convolutional modules was performed, employing the use of real-time image augmentation for data enrichment. All else being equal, data enrichment and fine-tuning each resulted in a 3-7% increase in classification accuracy in the best models. Future studies may explore more aggressive data enrichment techniques or perform deeper fine-tuning of the pre-trained MobileNet architecture. The list of models used for feature vector classification is by no means exhaustive, and many others may be tried in place of those used in this study. The portability of MobileNet may open the door to the possibility of in-field mobile classification, a topic for further discussion.

II. Data

A. Data

The dataset consists of 11588 images of 29 different classes of macroinvertebrates. The images provided had undergone cropping and centring of the subject matter. The dataset was split into 5830 training images, 2298 validation images and 3460 test images. The training and validation sets were levelled; the unlabelled test images were set aside for an ongoing classification competition. Several of the classes were of the same macroinvertebrate family; scrutinising between them was to prove difficult due to their similar traits. Furthermore, the training dataset was not perfectly balanced; the most represented class originally had 230 images, whilst the most underrepresented class had 114 images. Class imbalance can impart a bias within models such that underlying traits of the problem at hand are not actually learned. To address this, class expansion was used.

B. Enrichment

Typical training datasets for CNN's can run into the millions of images, this necessitated the enrichment of our relatively small dataset via image augmentation. The real-time image augmentation during CNN training was performed via the Keras API which provided a means of applying a random transformation to a given batch of images. The random transformation of each input image had a rotation range of 0-180 degrees, vertical and horizontal translation up to 15%, scale change up to 20% and skew up to 10%. As the augmentation process is random, the size of the new dataset during CNN training cannot be estimated. When generating feature vectors to be used in training the top model classifiers, images were augmented such that the classes were expanded to

approximately 1000 images each. Models such as SVM and PNN are not as heavily reliant on dataset size as, say, MLP. In the case of the former, their efficiency drops quickly with heavier enrichment. As such, when training these models, there is a trade-off between efficiency and accuracy. For example, training a SVM with a dataset of approximately 50,000 was intractable given the hardware at our disposal, however a dataset of approximately 15,000 required only a few minutes to train.

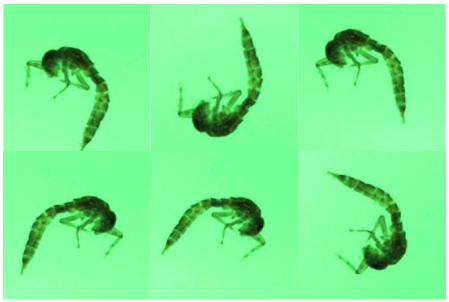

Figure 1: Six random augmentations of an image from the dataset

III. BASE MODEL

Transfer learning using large pre-trained CNN's has become common practice in recent years, particularly with regards to image classification [4]. Although initially thought only useful in the case of coarsely-grained classification tasks, such as cats vs. dogs, these large CNN's proved useful in fine-grained tasks, such as the one at hand. The convolutional modules of the network are able to extract unique and useful information regarding the subject matter's shape, shading and colour at multiple levels of abstraction.

A. *Architecture*

The pre-trained network used in this study was Google's MobileNet. MobileNet was created in an attempt to provide a low latency but accurate image classifier capable of being used in mobile and embedded vision applications. It employs depthwise separable convolutions that drastically reduce computational cost reportedly by a factor of 8-9 times [4]. A width factor of 1.0 was chosen, denoting the largest MobileNet structure. The input resolution was set as 224 pixels, again the largest value compatible with MobileNet.

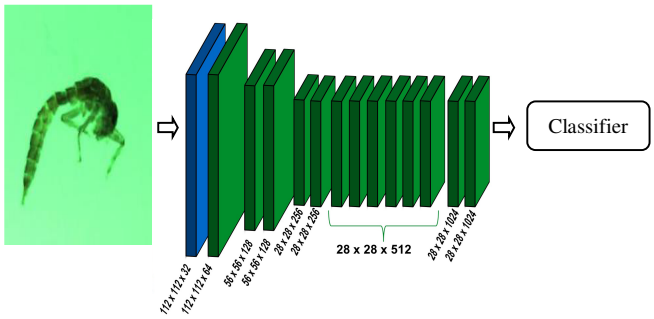

Figure 2: Final architecture with MobileNet and classifier [13]

The default output layer for final image classification is a single fully-connected MLP layer with 1000 output classes. For this study, the output layer is removed and replaced with one of the models proposed earlier. For fine-tuning of the base model, the top classifier was an MLP with a single hidden layer.

B. *Fine-tuning*

Although possible to use the bottleneck feature vectors generated by MobileNet with original weights, fine-tuning pre-trained models has been shown to improve performance [1]. The entropic capacity of the large MobileNet network means that simultaneously fine-tuning all convolutional modules would risk overfitting on our data [9]. Furthermore, as the earlier layers extract courser image detail than those layers farther downstream, fine-tuning these layers is not as pertinent in the case of fine-grained classification tasks. As such, only the last three convolutional modules were trained, alongside the top MLP module. To not induce large gradients, which would adversely disturb the weights of the convolutional blocks, the MLP is first trained on its own, with the rest of the network layers set as untrainable or 'frozen'. Next, the top-most convolutional module is 'unfrozen', and it, along with the MLP, is trained via stochastic gradient descent with a low learning rate and high momentum, again so not at to disturb the existing weights too much. This is done because the existing weights already capture some important image detail, and finetuning to our specific case does not require drastic changes. The process is then repeated, unfreezing the remaining modules chosen for fine-tuning.

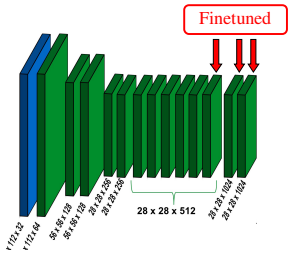

Figure 3: Convolutional modules for finetuning [13]

IV. CLASSIFIERS

Various top model classifiers were compared to decide which is the most effective for the task at hand. Four models were compared: multilayer perceptron network (MLP); probabilistic neural network (PNN); support vector machine (SVM); and extreme learning machine (ELM).

A. *Multilayer Perceptron Network*

A feedforward neural network was the first classifier used in conjunction with the base model's bottleneck feature vectors. A grid search was performed using a predetermined range for each of the network's hyperparameters. This included the number of hidden layers, the number of neurons in each layer,

dropout rate for the hidden layers, optimizing function, and activation function. The loss function to be optimized was the categorical cross-entropy function. Dropout is a technique used for regularization of network weights and prevents overfitting or overreliance on particular neurons. The dropout rate was set between 0.1-0.3 for the grid search. The optimizers tested consisted primarily of the Adam variations, with stochastic gradient descent and RMSProp also in the mix. Activation functions were ReLU, LeakyReLU, ELU, and tanh. An example of the grid search results is shown in table 1 which shows the top 5 accuracies for the enriched and fine-tuned model.

TABLE I. CLASSIFICATION ACCURACY GIVEN SET OF HYPERPARAMETERS FOR MLP NETWORK

No. Hidden Layers	*Neurons per layer*	*Dropout rate*	*Activation func.*	*Optimising func.*	*Ave. Acc. (%)*
1	400	0.3	Tanh	Adagrad	81.98
1	400	0.3	L-ReLU	Adagrad	81.65
1	400	0.2	Tanh	Adamax	81.56
1	800	0.3	ReLU	RMSprop	81.35
1	400	0.2	Tanh	Adagrad	81.07

B. Probabilistic Neural Network

Closely related to radial basis function networks, the probabilistic neural network is a feedforward neural network consisting of four layers: the input layer; hidden layer; summation layer; and output layer. The hidden layer, which comprises of one neuron for each case in the training dataset, computes the Euclidean distance of the test vector to each neuron's centre. The radial basis function is used as the activation function for the hidden layer, it's sigma value being a tunable hyperparameter. The value of sigma is crucial to the accuracy of the classification. Several sigma values were used, each within a magnitude of the input vectors range as suggested by the API documentation. The summation layer consists of one pattern neuron for each possible output class and is connected to the hidden layer by weighted paths. A path between a hidden layer neuron and a pattern neuron only exists if the actual class of the training case, represented by its hidden layer neuron, is the same as the class represented by the pattern neuron. Each pattern neuron sums its input values from the hidden layer and passes the value to the output layer, which identifies the class belonging to the largest pattern neuron output. Generally speaking, PNN's are faster than MLP's as they do not require training per se, they are also robust to outliers. As PNN's essentially store training cases and require distance calculations for each against the test case, they can be slower in classifying new test cases.

TABLE II. CLASSIFICATION ACCURACY (%) GIVEN SIGMA OF PNN

	Sigma			
Trial	*2*	*4*	*8*	*16*
1	71.22	69.32	54.22	48.21
2	70.14	64.76	51.80	48.45
3	71.52	61.47	50.93	44.70
4	68.51	61.97	54.48	49.13
5	69.70	54.22	54.48	48.13
Ave.	70.22	62.35	53.18	47.72

C. Support Vector Machine

A departure from the network-based machine learning models introduced so far, support vector machines employ the use of hyperplanes to separate the training data in two or more classes. It is often the case that the training data is not linearly separable, this necessitates the use of a kernel function to lift the data into a higher – and possibly infinite – dimension, where it is linearly separable. Hyperplanes are derived such that the margin between the closest – 'support' – vectors of each class are maximally spaced. In multi-class classification tasks, SVM's typically use either a one-v-one or one-v-rest algorithm. The former computes a decision boundary for each class pair, whereas the latter computes a decision boundary for each class which best separates it from the other classes. The API used defaulted to a one-v-one algorithm for multi-class classification. Although larger datasets may result in better decision boundaries, SVM's are most efficient with smaller datasets, in contrast with the data-hungry MLP. The hyperparameters to be selected were the weighting penalty 'C' and the kernel function. Prior knowledge of suitable ranges for these values were used for the hyperparameter grid search. The results of which can be found in table 3 for the enriched and fine-tuned model.

TABLE III. CLASSIFICATION ACCURACY (%) OF SVM GIVEN C VALUE AND KERNEL FUNCTION

C / *Kernel*	*0.001*	*0.01*	*0.1*	*1*	*10*	*100*	*1000*
RBF	7.05	28.54	69.49	78.98	82.16	81.81	79.76
Linear	77.46	79.63	78.59	79.15	80.11	79.11	79.11

D. Extreme Learning Machine

Another feedforward network-based machine learning model, the extreme learning machine is essentially an MLP without backpropagation in that the weights to the hidden layer(s) are constant. Only weights connecting the final hidden layer to the output layer are updated, in a single step, which equates to training a linear regressor/classifier. Parameters associated with hidden nodes also do not require tuning. As one may expect, this results in much shorter training times compared to backpropagation based MLP. Literature suggest ELM can

outperform many other models, despite the apparent randomness of the input weights and neuron parameters [10]. ELM has experienced much research in the past decade and the performance of its models can be among the best [11] [12]. The ELM architecture used in this study comprised of a single hidden layer network with a weighted combination of radial basis function (RBF) activation and MLP activation function, and a fully-connected output layer. The number of MLP activation function, number of hidden neurons, width of the radial basis function, and the mixing parameter (alpha) – that controls the contribution of both the RBF activation and MLP activation – are tuned for optimal performance.

TABLE IV. CLASSIFICATION ACCURACY (%) GIVEN SET OF HYPERPARAMETERS FOR ELM

Neurons	*Alpha*	*Activation func.*	*RBF Width*	*Ave. Acc.*
400	0.2	Tanh	0.6	41.25
200	1	Tanh	6	40.76
400	0.2	Sigmoid	0.6	39.88
400	0.2	Tanh	0.6	39.87
400	1	Tanh	6	38.98

V. EXPERIMENTAL RESULTS

The experimental process involved five stages. The first investigated the effect of data enrichment on classification accuracy of each top model classifier. The second looked at the effects of fine-tuning the base model. The third compared all combinations of enrichment and fine-tuning. The fourth took a closer look at the classification metrics for individual classes and models, this provides important information at to where the model is performing well or, conversely, poorly. The last section of the experiments regards the classification of the test cases, and the accuracy achieved for the ongoing Kaggle competition.

A. Data Enrichment

Table 5 shows the classification accuracy on the validation dataset for models trained both on the original training set and its enriched counterpart. The approximately optimal hyperparameters of each model were found prior to recording final accuracy scores. Scores were averaged over 5 trial runs.

TABLE V. CLASSIFICATION ACCURACY USING ENRICHMENT

	MLP		*SVM*		*PNN*		*ELM*	
Trial	*Enrich*	*Orig.*	*Enrich*	*Orig.*	*Enrich*	*Orig.*	*Enrich*	*Orig.*
1	77.55	76.21	77.40	75.78	68.95	66.31	38.75	38.65
2	77.49	75.44	76.38	75.90	67.66	66.90	39.55	39.11
3	77.16	75.90	76.80	76.01	65.42	64.47	39.43	39.54
4	77.68	74.31	76.52	76.07	67.95	67.67	37.89	37.39
5	76.87	75.01	77.11	75.22	69.24	68.21	39.54	39.09
Ave.	77.35	75.37	76.84	75.80	67.84	66.71	39.02	38.76

B. Fine-tuning

Table 6 shows the classification accuracy on the validation dataset for models using both the original base model feature vectors and the fine-tuned base model feature vectors. Again, the scores were averaged over 5 trial runs. The datasets were not enriched.

TABLE VI. CLASSIFICATION ACCURACY (%) USING FINETUNING

	MLP		*SVM*		*PNN*		*ELM*	
Trial	*Fine*	*Orig.*	*Fine*	*Orig.*	*Fine*	*Orig.*	*Fine*	*Orig.*
1	79.19	76.44	81.11	76.00	71.11	68.45	41.12	38.76
2	79.06	75.98	81.04	76.22	70.23	66.90	41.11	39.90
3	81.07	75.78	81.54	75.53	71.11	68.41	41.18	38.42
4	79.65	76.32	81.32	76.12	69.96	64.05	40.09	38.99
5	79.86	75.01	80.87	76.77	70.23	67.33	40.87	38.89
Ave.	79.77	75.91	81.18	76.13	70.53	67.03	40.87	38.99

C. Enrichment and Fine-tuning

Table 7 shows the average classification accuracies on the validation dataset for the original dataset with and without fine-tuning, and the enriched dataset with and without fine-tuning. The models use the best hyperparameters found in section 4.

TABLE VII. CLASSIFICATION ACCURACY (%) ON VALIDATION DATASET

Model	*Original*	*Original +Finetune*	*Enriched*	*Enriched +Finetune*
MLP	75.64	79.77	77.35	81.98
SVM	75.97	81.18	76.84	82.16
PNN	66.87	70.53	67.84	70.22
ELM	38.88	40.87	39.02	41.25

D. Per Class Classification Scores

Table 8 shows the precision, recall and F-score of each class for the MLP trained using the enriched dataset of finetuned feature vectors. The best of the other models produced similarly distributed values per class.

TABLE VIII. CLASSIFICATION REPORT

Class	*Precision*	*Recall*	*F-Score*	*Class*	*Precision*	*Recall*	*F-Score*
1	0.92	0.95	0.94	***16***	0.80	0.82	0.81
2	0.62	0.73	0.67	***17***	0.99	0.96	0.97
3	0.79	0.78	0.78	***18***	0.84	0.93	0.88
4	0.90	0.88	0.89	***19***	0.96	0.92	0.94
5	0.87	0.85	0.86	***20***	0.71	0.55	0.62
6	0.64	0.63	0.63	***21***	0.94	0.96	0.95
7	0.65	0.63	0.64	***22***	0.64	0.78	0.70
8	0.65	0.55	0.59	***23***	0.88	0.86	0.87
9	0.86	0.91	0.88	***24***	0.79	0.79	0.79
10	0.83	0.82	0.82	***25***	0.50	0.47	0.48
11	0.92	0.95	0.94	***26***	0.85	0.83	0.81
12	0.86	0.85	0.86	***27***	0.97	1.00	0.98
13	0.91	0.99	0.95	***28***	0.80	0.83	0.81
14	0.84	0.80	0.82	***29***	0.90	0.70	0.79
15	0.75	0.73	0.74				

E. Classification of Test Dataset

Table 9 shows the top classification accuracies achieved by each model on the test data held out for the ongoing Kaggle competition. The models were trained using the enriched dataset with fine-tuning.

TABLE IX. CLASSIFICATION ACCURACY (%) ON TEST DATASET

Trial	***MLP***	***SVM***	***PNN***	***ELM***
1	81.41	82.43	71.11	41.12
2	81.33	81.91	69.96	38.90
3	81.21	81.33	-	-

VI. DISCUSSION

Data enrichment alone resulted in a 1-2% increase in classification accuracy, although the size of the dataset adversely affected the training speed of the SVM, PNN and ELM with slightly smaller return in accuracy compared to the MLP. Further augmentation of input images may result in higher accuracy for the MLP but the same remains to be seen for the SVM, PNN and ELM models. Overall, the gains seen from image augmentation were underwhelming across all models. Finetuning, also a time-consuming process, resulted in only a 2-5% increase in classification accuracy, all else being equal. As the quality of feature vector representation increases with fine-tuning of the base model, unfreezing more convolutional modules during training may result in further gains in accuracy. Furthermore, the base model was only finetuned for 70 epochs, quite a short amount of time when using stochastic gradient descent with a low learning rate as the optimizing algorithm. Extending this beyond 100 epochs, with careful consideration not to overfit the data, could also result in increased accuracy. From part D of the experimental results, it is clear that not all classes are considered equal when it comes to precision and recall. Although many classes achieve above 90% precision, many fall below 60%. Class 2, the second largest support in the original dataset, was classified quite poorly, ~62% precision at best, and would account for a large portion of the misclassified samples in the validation and test results. Likewise, other well represented and low precision classes, such as 6,7 and 8, also would also affect classification accuracy more so than those classes with small support and low accuracy. Evidently, these classes of macroinvertebrate are not particularly distinct from the rest, and the models are less confident in their classification. Methods for separating these classes are for future studies. Overall, the SVM was the model of choice, which one may expect given the relatively small dataset originally presented. The MLP was a close second, with data enrichment boosting its results by the most compared to the other models. The PNN and ELM were a distant third and last, a surprising result considering the praise each receive in many publications. Neither enrichment nor finetuning seemed to improve the accuracy of the latter two models by a significant margin.

VII. CONCLUSION

The validity of adopting transfer learning to generate learned features for fine-grained multi-class classification appears to be sound. That being said, past studies have shown that simpler geometrical and intensity-based feature vectors achieve better results. Future studies on the subject may apply deeper and longer finetuning of the base model, more aggressive image augmentation, and investigate new methods of separating non-distinct classes.

REFERENCES

[1] E. Riabchenko, K. Meissner, I. Ahmad, A. Iosifidis, V. Tirronen, M. Gabbouj, and S. Kiranyaz, “Learned vs. engineered features for fine-grained classification of aquatic macroinvertebrates,” in Accepted to International Conference on Pattern Recognition (ICPR), 2016.

[2] J. Raitoharju, E. Riabchenko, K. Meissner, I. Ahmad, A. Iosifidis, M. Gabbouj and S. Kiranyaz, "Data Enrichment in Fine-Grained Classification of Aquatic Macroinvertebrates", *2016 ICPR 2nd Workshop on Computer Vision for Analysis of Underwater Imagery (CVAUI)*, 2016.

[3] H. Joutsijoki, K. Meissner, M. Gabbouj, S. Kiranyaz, J. Raitoharju, J. Ärje, S. Kärkkäinen, V. Tirronen, T. Turpeinen and M. Juhola, "Evaluating the performance of artificial neural networks for the classification of freshwater benthic macroinvertebrates", *Ecological Informatics*, vol. 20, pp. 1-12, 2014.

[4] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", *arXiv preprint arXiv:1704.04861v1*, 2017.

[5] H. Joutsijoki and M. Juhola, "A comparison of classification methods in automated taxa identification of benthic macroinvertebrates", *International Journal of Data Science*, vol. 2, no. 4, p. 273, 2017.

[6] D. Stathakis, "How many hidden layers and nodes?", *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133-2147, 2009.

[7] F. Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357v2, 2016.

[8] J. Arje, V. Tirronon, S. Karkkainen, K. Meissner, J. Raitoharju, A. Iosifidis, M. Gabbouj and S. Kiranyaz, "Human Experts vs. Machine in Taxa Recognition", *arXiv preprint arXiv:1708.06899v2*, 2017.

[9] F. Chollet, "Building powerful image classification models using very little data", *Blog.keras.io*, 2016. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html. [Accessed: 10- May- 2018].

[10] G. Huang, Q. Zhu and C. Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, vol. 70, no. 1-3, pp. 489-501, 2006.

[11] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding and Rui Zhang, "Extreme Learning Machine for Regression and Multiclass Classification", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513-529, 2012.

[12] G. Huang, "What are Extreme Learning Machines? Filling the Gap Between Frank Rosenblatt’s Dream and John von Neumann’s Puzzle", *Cognitive Computation*, vol. 7, no. 3, pp. 263-278, 2015.

[13] A. Levinshtein, C. Chang, E. Phung, I. Kezele, W. Guo and P. Aarabi, "Real-time deep hair matting on mobile devices", *arXiv preprint, arXiv:1712.07168*, 2018.