2021 36th International Conference on Image and Vision Computing New Zealand (IVCNZ) | 978-1-6654-0645-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/IVCNZ54163.2021.9653412

Deep Convolutional Neural Networks for Fish Weight Prediction from Images

Yunhan Yang[1], Bing Xue[1], Linley Jesson[2], Matthew Wylie[3], Mengjie Zhang[1] and Maren Wellenreuther[3,4]
[1]*School of Engineering and Computer Science, Victoria University of Wellington*, Wellington, New Zealand
[2]*The New Zealand Institute for Plant and Food Research Limited*, Auckland, New Zealand
[3]*The New Zealand Institute for Plant and Food Research Limited*, Nelson, New Zealand
[4]*School of Biological Sciences, The University of Auckland*, Auckland, New Zealand
Email: {yangyunh, bing.xue, mengjie.zhang}@ecs.vuw.ac.nz, and
{linley.jesson, matthew.wylie, maren.wellenreuther}@plantandfood.co.nz

Abstract—**Fish weight is an important performance trait in aquaculture, conservation, fisheries science and management since weight relates to the growth of individual fish in a particular environment. A power regression model is commonly used to explain the relationship between fish weight and length. However, this requires costly measurements of fish length. The present study applies machine learning techniques to predict fish weight from fish images, bypassing the length measurement step. In this study, we validate the feasibility of predicting fish weight from images directly. We use a convolutional neural networks (CNNs) based approach to predict fish weight from images by building regression models. The deep CNNs architecture VGG-11, ResNet-18 and DenseNet-121 are chosen to train the models. The fish images have different scales (length-pixel ratio) without including a ruler as a reference. The trained regressors of these three architectures reach R^2 0.94, 0.95 and 0.96 on the test set. Our results support the feasibility of fish weight prediction with the CNNs model from images directly. The fish images look similar to humans, but CNNs regressors can detect the different fish weights. The CNNs regressors also can detect the fish images with different length-pixel ratios.**

I. Introduction

Fish weight provides important information for aquaculture, conservation, fisheries science and management. Specifically, data on fish weights provide insights into fish growth and health over a period of time, the environmental impacts of habitats on fish weight across populations, and the nutritional quality and consumption of different diets by fish [1].

To predict fish weight, researchers focus on the relationship between fish length and fish weight, termed here fish length-weight relationship (LWR). A power function is commonly used to explain the LWR: $W = aL^b$, which uses the fish length (L in cm) to predict the fish weight (W in grams). The intercept coefficient a and the exponential b terms vary according to the fish species and the growth conditions. Different measures of elongation (e.g. total length, fork length, body height) can be used depending on the shape of the fish [1], [2]. However, manual measuring is needed for the fish lengths to build the model.

Machine learning algorithms have been widely applied to aquaculture and fisheries science to shorten the manual measuring process. As an emerging machine learning technique, convolutional neural networks (CNNs) have proved their capabilities on computer vision tasks such as image classification, segmentation, etc. Many studies use CNNs to predict fish weight. The studies [3], and [4] use image segmentation and masking to extract morphological features of the fish, which also makes the weight prediction more accurate. These studies either use a ruler as a reference of the fish scale (length-pixel ratio) or use the images with a fixed scale taken by benchtop units in fixed height. Several advanced approaches [5], [6] extract morphological features from images taken by several underwater cameras. These approaches achieve very high prediction accuracy and minimize fish stress and handling during fish sampling events and reduce human effort considerably.

978-1-6654-0645-1/21/$31.00 ©2021 IEEE

In this study, we are investigating the feasibility to use CNNs to predict fish weight directly from fish images. The CNNs models are expected to extract features automatically and skip the manual measuring process.

This work aims to validate the feasibility of predicting fish weight from images taken with benchtop or underwater cameras without using a ruler or fixing the scales. Specifically, we will

- train deep CNNs models to predict fish weight from images;
- evaluate the performance of the CNNs regressors on the fish weight prediction; and
- analyze the prediction results from images.

II. Background

A. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are typically used for supervised learning that form the basis for computer vision tasks such as image classification, localization, and segmentation [7]. It was initially proposed in the late 1980s [8], was inspired by "Neocognitron" in the early 1980 [9]. A CNN architecture is very similar to a neural network but with some variations such as the attachment of convolutional layers, max-pooling layers, etc.

AlexNet (shown in Fig. 1) is a modern deep CNNs architecture. The first layer is the image as the input layer. It is

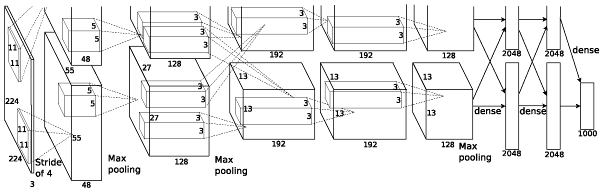

Fig. 1. AlexNet - a modern deep CNNs architecture published in 2012 [10], see text for description.

represented as a matrix with 224×224 pixels and three color channels (RGB). Five convolutional layers follow the input layer. Each of the three max-pooling layers follows the 1st, 2nd and 5th convolutional layers. The last three layers are the fully connected neural network layers. The convolutional layers are used to extract the features from the input images. The matrix of 11×11 on the input layer is a convolutional filter, which consists of 11×11 trainable weights. The filter convolves on the previous layer (the previous layer can be the input image or a convolutional layer) and extracts the features to form a feature map. The feature map then applies an activation function (e.g. $Sigmoid$, $tanh$, $ReLU$, etc.), which forms the output of the convolutional layer. In addition, the hidden units on the convolutional layer are the extracted features in the feature map; they all share the same weights of the convolutional filter. Each hidden unit in a feature map is connected to the feature map of the previous layer via the trainable weights of the filter [7], For AlexNet, it contains 32 filters $\times$ 3 color channels (= 96 filters) in the first convolutional layer.

A max-pooling layer applies the operation that selects the largest element (downsampling) in each predefined region of the previous feature map. It is used to reduce the spatial resolution of the feature maps without losing any information. The reduction in the number of parameters also improves the computational efficiency [7].

The fully connected layers follow the convolutional and max-pooling layers to interpret the extracted features and perform the function of high-level reasoning [7]. The outputs from convolutional and max-pooling layers are flattened and fed into the fully connected layers. The structure of the fully connected layers is the same as a common neural network. A softmax activation function is applied on the last layer to solve a classification problem. AlexNet has 1,000 nodes in the last layer, which is for the classification with 1,000 categories.

The training processing of CNNs is the same as a neural network. The weights of the filters in the convolutional layers are trained during backpropagation. For a max-pooling layer, the gradient from the previous layer is only for the largest element in the region. Then it back-propagates the gradient to the convolutional layer. Other elements in the region have zero gradients.

The design/search of the deep CNNs architecture is still a popular topic and being improved rapidly. There are several representative architectures such as:

- VGG [11] - the architecture is based on the AlexNet, but able to include more weight layers to make it deeper using smaller convolution filters. Also, it introduces the concept of building blocks containing a list of layers with the same structure;
- ResNet (Residual Network) [12] - the stacking of the neural network layers makes the model more difficult to be optimized/trained. ResNet introduces the residual connections, which allows the input of a building block to be propagated to the deeper block directly. It also reduces the model complexity;
- DenseNet [13] - similar to ResNet. In each dense block, the input of a convolutional layer is the concatenation of its preceding layers. The architecture reduces the complexity, solves the vanishing gradients problem, and makes the weights training more efficient.

The traditional deep CNNs architectures such as AlexNet and VGG have a large number of trainable parameters. The recent architectures (e.g. ResNet and DenseNet) are trained with fewer trainable parameters, and have a better classification accuracy rate.

There are some training techniques to avoid overfitting, such as ridge (L2) regularization, dropout layer [14]. The batch normalization (BN) [15] helps to avoid vanishing/exploding gradients and make the model more concrete on different distributions of the dataset.

B. Predict Fish Weight from Images

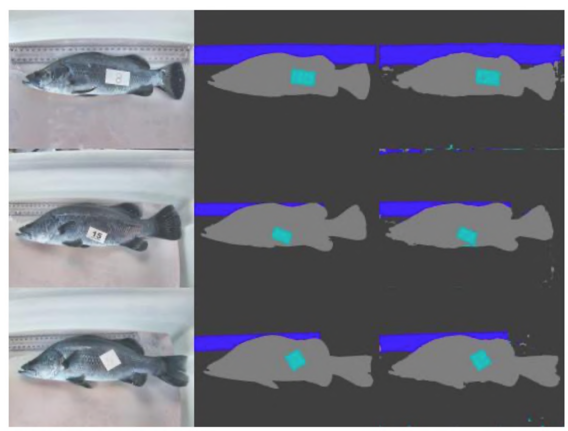

Fig. 2. An example of the image segmentation [16]

For the measurement of fish weight or morphological features from images, many recent studies use image segmentation [3], [4], [5], [6], [16]. Image segmentation recognizes the objects on the image. These are generally using different learning and optimization algorithms, including neural networks and evolutionary learning [17], [18]. Fig. 2 [16] gives an example of using image segmentation to distinguish a ruler (i.e. a scale bar), a label (to distinguish individual fish) and an individual fish from the background. The segmentation is used to scale the fish image automatically. The blue area is segmented as the ruler, and the gray area is segmented as the body of the fish.

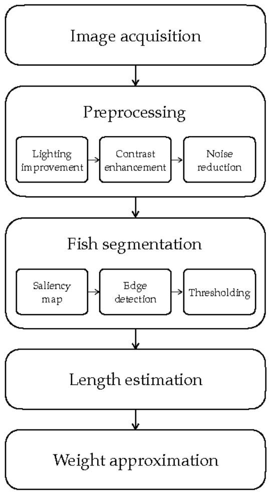

Fig. 3. An example of the process of fish weight prediction from underwater fish images using a power function [5]

Fig. 3 [5] shows an example of the process to predict fish weight from underwater fish images. The latter study introduces an approach to measure the fish weight with underwater cameras so that no live fish extraction is involved. This approach reduces fish stress due to eliminating prolonged exposure out of the water. Hatchery managers can also access feeding information and are able to simultaneously measure other population parameters. Image pre-processing is needed for the segmentation process. It includes exposure and contrast adjustment, and noise-canceling. The image segmentation algorithm is conducted with a combination of homomorphic filtering, contrast limited adaptive histogram equalization (CLAHE) and guided filter. Lastly, the length metrics are extracted to measure the fish with a power function.

More recent studies use CNNs to achieve the image segmentation, such as [19] extracts the area of the fish body with image segmentation and predicts the fish weight. The studies [3], [4] use a mask region-based CNNs (Mask R-CNNs) [20] to extract the morphological features.

III. Proposed Methods

A. CNNs Architecture

In this work, we choose three representative CNNs architectures to train regression models to predict fish weight:

- VGG-11 with Batch Normalization: a traditional deep CNNs architecture that stacks the weight layers;
- ResNet-18: a CNNs architecture introduces the concept of residual block;
- DenseNet-121: more advanced CNNs architecture allows to pass the input to deep layers.

These architectures are built in the `PyTorch` library [21] with pre-trained parameters. However, the initial experiments show that the models with pre-trained parameters have poor performance on fish weight prediction. The main reason is that the parameters were originally pre-trained for the classification tasks, but this does not apply to the regression task. To cope with fish weight prediction, which is a regression task, we need to use a linear layer as the model's output. The output layer outputs a single element for the predicted weight given an image. There is no need to set the activation function for the regression task.

B. Training Pipeline

The main steps to train the regressors:

- Pre-process the dataset (we discuss the dataset and the data pre-processing step in the next section);
- Load the pre-trained architecture in `PyTorch`;
- Configure the output layer for a regression task;
- The batch size is set to 32;
- Set $Adam$ [22] as the optimizer of CNNs. The learning rate is set to 0.001, which is tuned to shorten the training time;
- Set $Mean\ Squared\ Error$ as the loss function, which is common for a regression problem;
- Train the models with different CNNs architectures. Each training runs with 100 epochs, since it shows the training is able to converge within 100 epochs.

C. Training Environment

The `Python` library `PyTorch` is used for training the CNNs regressors. The experiments are run on the high-performance computing (HPC) facility at The New Zealand Institute for Plant and Food Research Limited (PFR). The HPC equips an NVIDIA Tesla P100 GPU (Graphics Processing Unit) with 16 GB memory. It utilizes the GPU acceleration for the training while `PyTorch` supports NVIDIA's CUDA [23] APIs.

IV. Experiment Design

A. Dataset

The dataset consisted of 259 Australasian snapper (*Chrysophrys auratus*, tāmure in Māori; hereafter referred to as snapper) individuals of which a total of 529 images were collected from these (i.e. some replicate images of the same individual). The dataset was provided by PFR. One to three photos were taken for each fish individual, which ensures at least one image is clear and in focus.

Images were collected from captive-bred snapper produced and held at the Nelson Research Centre finfish facility, New Zealand. As part of a wider study to describe the reproductive biology of snapper in captivity, two-year-old juveniles were maintained under ambient flow-through water temperature and photoperiod conditions in a 5,000 L tank. Fish were fed daily by hand to satiation on a diet consisting of commercial pellet feeds (Skretting and/or Ridley) supplemented with frozen squid (*Nototodarus spp.*) and an in-house mixed seafood diet enriched with vitamins. For this study, sampling commenced in September 2019. More samples were collected periodically every 4 to 6 or 12 weeks until June 2021. At each sampling point, fish were subjected to complete sedation and euthanasia by overdose in anesthetic ($>$ 50 ppm AQUI-S®; Aqui-S New Zealand Ltd, Lower Hutt, New Zealand) before being photographed. Fig. 4 demonstrates some examples of the fish

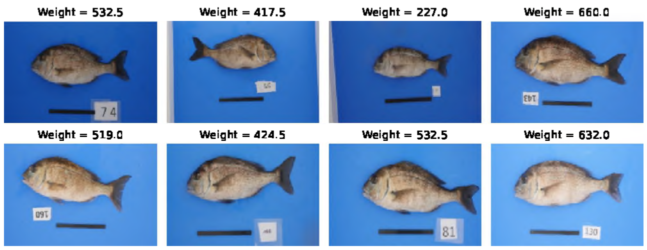

Fig. 4. Examples of original fish images used in this study to predict weight.

images. Subsequently, the body morphometrics (body weight and fork length) of each individual were measured manually before fish were dissected to collect a range of tissues to confirm the sex of each individual and to record traits and collect samples for other research purposes. The animal handling and manipulations were approved and conducted according to the guidelines of the Nelson Marlborough Institute of Technology Animal Ethics Committee.

The fish cover a range of 214.0 to 1280.0 grams in weight. From the example images of Fig. 4, we notice that the size of the fish in the image is not associated with its weight, because the images are not scaled to match a fixed length-pixel ratio. A ruler with a scale bar is placed to give a length reference. However, we remove the ruler to examine whether CNNs can predict the weight without scale information provided by the ruler. The scenario of measuring without a length reference or a fixed scale is also closer to the weight prediction from underwater images.

B. *Training and Test Datasets*

We use 5-fold cross-validation (CV) to assess the overall performance of each CNNs architecture. The same CV folds are used to train five regression models of the three architectures. The best model with the lowest prediction error is selected to measure the architecture's performance of the CV fold. We average the performance on all the folds to measure the overall performance of the architecture.

Multiple images were taken for each fish individual. To ensure the independence of training and test sets, it is important to avoid the images of the same fish individual appearing in both training and test sets. There are around 420 instances in the training set and about 110 instances in the test set.

C. *Image Pre-processing*

The following pre-processing steps are needed before feeding the image into the CNNs regressors:

- Crop the ruler by removing the area of the ruler at the bottom;
- Pad the image to make a square image;
- Resize the image to (224×224) pixels to fit the input size of the pre-defined CNNs architectures;
- Set to flip the image horizontally and vertically randomly. The probability of the flip is 0.5;
- Normalize the images with `[0.485, 0.456, 0.406]` mean and `[0.229, 0.224, 0.225]` standard deviation.

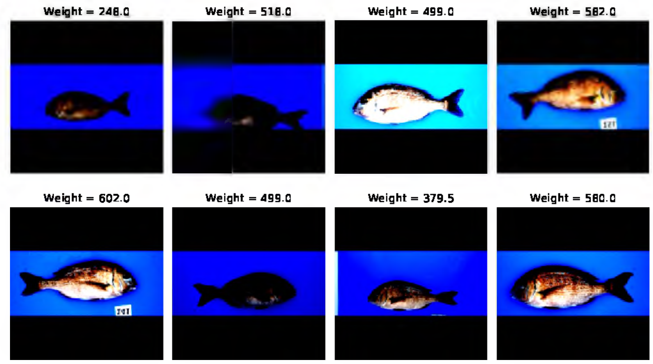

Fig. 5. Examples of the pre-processed images used in this study to predict fish weight. The ruler was removed from the picture during pre-processing to assess whether CNNs could predict weight in the absence of a scale.

Fig. 5 presents the examples of pre-processed images. The values on RGB are normalized so that the images look unusual.

The random flip data augmentation enlarges the dataset and helps to prevent overfitting since the CNNs model gets trained with more fish images with different augmentations but the same weight. The data augmentation applies only to the training process. To the test set, only cropping, padding, resizing and normalization transforms are applied.

D. *Measurements*

We use R^2 to measure the performance of the architectures, which indicates the proportion of the explained variation of the model. In addition, the *Mean Squared Error* (MSE) and *Root Mean Squared Error* ($RMSE$) are used to evaluate the difference between the predicted fish weight and actual fish weight.

The training process is time-consuming: the training of each CNNs regressor takes about 5.5 to 8 hours (330 to 480 minutes) for running 100 epochs on each CV fold. We train each model of the three architectures with 5 CV folds once only. We compare the best performance (the best MSE, $RMSE$ and R^2 in 100 epochs of each CV fold) and the average performance (the average of the best performance of 5 CV folds) among these CNNs architectures, look into the learning curves for these training processes, and analyze the fish weight prediction.

V. Results and Analysis

This section discusses the experimental results and analysis. Table I shows the best and the average MSE, $RMSE$ and R^2 on the test sets. Fig. 6 shows the learning curves of the training process. Fig. 7 compares the predicted weights by the regressors with the actual weights. Fig. 8 presents some examples of fish weight prediction using images.

Table I presents the VGG-11, ResNet-18 and DenseNet-121 models with the best MSE, $RMSE$ and R^2 on the test set, each CV fold and the average results of all 5 CV folds within 100 epochs. We observe the CV folds have different performances on the three CNNs architectures: the CV1 (1st fold) and CV3 perform better on the test set than

TABLE I
THE BEST AND AVERAGE MSE, $RMSE$ AND R^2 ON THE TEST SET IN 100 EPOCHS

	Test MSE			Test $RMSE$			Test R^2		
	VGG	ResNet	DenseNet	VGG	ResNet	DenseNet	VGG	ResNet	DenseNet
CV1	3067.61	2369.19	1788.04	55.39	48.67	42.29	0.96	0.97	0.98
CV2	5016.45	3039.54	2823.26	70.83	55.13	53.13	0.93	0.96	0.96
CV3	5005.40	2890.42	2576.58	70.75	53.76	50.76	0.94	0.97	0.97
CV4	5271.63	4023.04	3551.07	72.61	63.43	59.59	0.92	0.94	0.95
CV5	4037.06	5965.43	3380.03	63.54	77.24	58.14	0.94	0.91	0.95
Average	4474.48	3620.89	2818.11	66.89	60.17	53.09	0.94	0.95	0.96

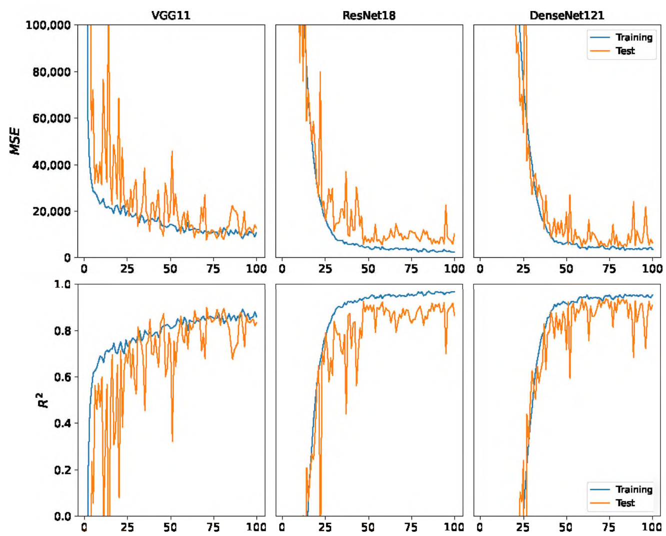

Fig. 6. Learning curves for the training of the three architectures

other folds. The average value provides a general view of the performance of the architectures on the dataset. Among the CNNs architectures, DenseNet-121 has the highest R^2 (0.96): better than ResNet-18 (0.95) and VGG-11 (0.94). Regarding $RMSE$, every prediction by DenseNet-121 has an average of 53.09 grams error, which is lower compared with ResNet-18: an average of 60.17 grams error, and VGG-11: an average of 66.89 grams error on the dataset. The performance on each CV fold has a consistent pattern: DenseNet-121 has a lower test MSE and $RMSE$, and a higher test set R^2 than the other two architectures.

Fig. 6 shows the learning curves of the three architectures: the average MSE and R^2 of 5 CV folds over the 100 epochs. There is no sign of underfitting or overfitting on the VGG-11 and ResNet-18, except there is a slightly overfitting on DenseNet-121 after the 75th epoch. The trending of average MSE and R^2 on the training set is very smooth. However, the curves are very wobbly on the test set, especially the first 25 epochs of the VGG-11 model. The wobbliness gets less frequent after the 50th epoch among the three models. The ResNet-18 model has better performance on the training set than the DenseNet-121's training set performance. It has a larger variance between the training set and the test set, which makes the performance on the test set is not as good as the DenseNet-121's test set performance.

According to Fig. 7: the fish weights prediction by three CNNs regressors look very similar: the residuals (i.e. prediction errors) spread evenly across the line of actual weights.

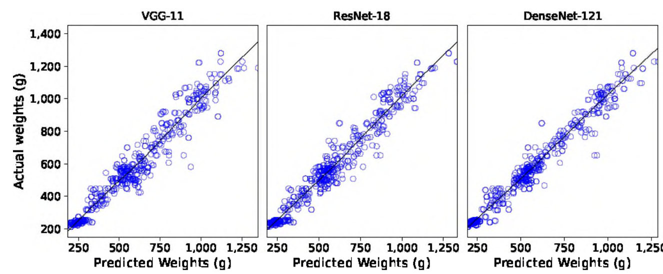

Fig. 7. The actual vs. predicted weights from the aggregation of the prediction on the test sets of each CV fold predicted by the best models.

There is no sign of heteroscedasticity where the heavier fish does not cause more prediction errors. The DenseNet-121 does perform better than the other two models with fewer prediction errors, which has an excellent prediction for the fish weighed from 650g to 950g. The ResNet-18 model performs better for the larger fish ($>$ 1,100g). All three CNNs regressors have a small number of outliers.

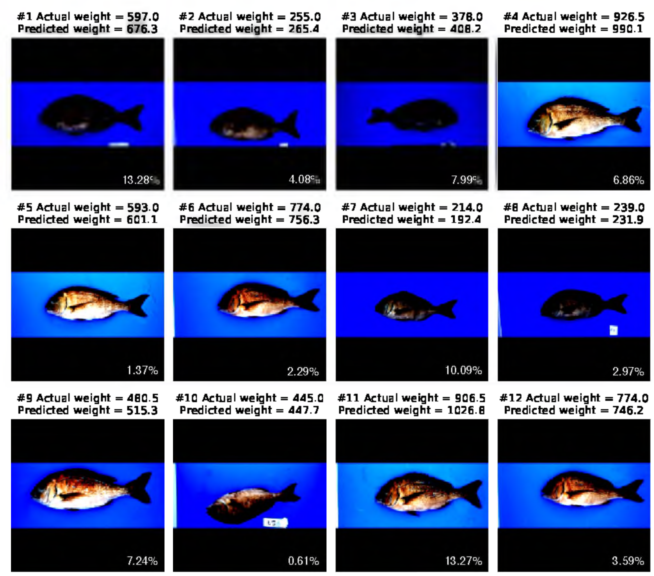

Fig. 8. Examples of fish weight prediction on the test set using the best DenseNet-121 model trained by CV1.

Fig. 8 demonstrates the fish weight prediction by DenseNet-121 regressor on the test set of CV1 for it has the highest R^2. The figures at the right bottom of each image indicate the prediction error ratios based on their actual weights ($prediction\ error/actual\ weight$). Most predictions have very small errors except #1, #4 and #11 ($>$ 50 grams). This suggests that the regressor is capable of recognizing the difference of image scales without inclusion of a ruler. For example, the fish in image #9 appears a larger size than in images #5, #6 and #12, but the regressor can recognize the individual has a lighter weight. For the predictions with larger errors: in images #1, #4 and #11, the error ratios are at a relatively low level according to their actual weights.

VI. Conclusions and Future Work

The CNNs based approach was applied to a snapper image set to predict the fish weight. We compared the performance of three deep CNNs architectures. The architectures had similar performances, and all showed a high predictive ability for fish weight. It supported the feasibility of using CNNs for fish weight prediction from images directly. The CNNs regressors trained without using a ruler and a fixed scale also provided good predictive power and showed its capability on the fish weight prediction.

We also observed some predictions with errors. One improvement is to pre-process the fish images so they have the same scale before predicting fish weight. The study [24] introduces a method to scale the image with a ruler. With the scaled images in a fixed pixels-per-millimeter ratio, it is worth checking if the CNNs regressor can achieve even higher fish weight prediction performance. Another future direction could be the interpretation/explanation of the CNNs regressors. The lack of interpretation/explanation always makes the prediction risky to be applied in practice. Studies such as Grad-CAM [25], which try to interpret the target class gradients of the final convolutional layer, provide some degree of explanation to highlight the important regions of the prediction. Although, the model is for a classification problem, it is worth looking into the methods of how it can be applied to a regression problem.

Acknowledgments

This work was funded in part by the Ministry of Business, Innovation and Employment Data Science SSIF Fund under Grant RTVU1914, New Zealand Government via the Ngā Pou Rangahau platform, a research framework developed by The New Zealand Institute for Plant and Food Research Limited, the Science for Technological Innovation Challenge under Grant 2019-S7-CRS, the Marsden Fund of New Zealand Government under Grant MFP-VUW1913 and MFP-VUW1914, and the University Research Fund at Victoria University of Wellington (grant number 223805/3986).

References

[1] R. Froese, "Cube law, condition factor and weight-length relationships: history, meta-analysis and recommendations," *Journal of Applied Ichthyology*, vol. 22, no. 4, pp. 241–253, Aug. 2006.

[2] Y. Yang, B. Xue, L. Jesson, and M. Zhang, "Genetic programming for symbolic regression: A study on fish weight prediction," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 588–595.

[3] C. Yu, X. Fan, Z. Hu, X. Xia, Y. Zhao, R. Li, and Y. Bai, "Segmentation and measurement scheme for fish morphological features based on mask r-cnn," *Information Processing in Agriculture*, vol. 7, no. 4, pp. 523–534, 2020.

[4] N. Bravata, D. Kelly, J. Eickholt, J. Bryan, S. Miehls, and D. Zielinski, "Applications of deep convolutional neural networks to predict length, circumference, and weight from mostly dewatered images of fish," *Ecology and Evolution*, vol. 10, pp. 9313 – 9325, 2020.

[5] G. Sanchez, A. Ceballos Arroyo, and S. Robles-Serrano, "Automatic measurement of fish weight and size by processing underwater hatchery images," *Engineering Letters*, vol. 26, pp. 461–472, 01 2018.

[6] N. Petrellis, "Measurement of fish morphological features through image processing and deep learning techniques," *Applied Sciences*, vol. 11, p. 4416, 05 2021.

[7] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, pp. 1–98, 06 2017.

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[9] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[13] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[16] D. A. Konovalov, J. A. Domingos, R. D. White, and D. R. Jerry, "Automatic scaling of fish images," in *Proceedings of the 2nd International Conference on Advances in Image Processing*, ser. ICAIP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 48–53.

[17] M. Setayesh, M. Zhang, and M. Johnston, "A novel particle swarm optimisation approach to detecting continuous, thin and smooth edges in noisy images," *Information Sciences: an International Journal*, vol. 246, pp. 28–51, 10 2013.

[18] Y. Liang, M. Zhang, and W. Browne, "Figure-ground image segmentation using feature-based multi-objective genetic programming techniques," *Neural Computing and Applications*, vol. 31, p. 3075–3094, 07 2019.

[19] D. A. Konovalov, A. Saleh, D. B. Efremova, J. A. Domingos, and D. R. Jerry, "Automatic weight estimation of harvested fish from images," 2019.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[23] S. Cook, *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012.

[24] D. A. Konovalov, J. A. Domingos, C. Bajema, R. D. White, and D. R. Jerry, "Ruler detection for automatic scaling of fish images," in *Proceedings of the International Conference on Advances in Image Processing*, ser. ICAIP 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 90–95.

[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct 2019.