Underwater Fish Species Recognition Model and Real-time Recognition System (Postprint)

Authors: Li Shaobo, Yang Ling, Yu Huihui, Chen Yingyi

Date: 2023-02-17T00:00:00+00:00

Abstract

Rapid and accurate fish recognition systems require robust recognition models and deployment systems as support. In recent years, convolutional neural networks have achieved tremendous success in the field of image recognition. Different convolutional network models possess distinct advantages and disadvantages, and faced with numerous optional model architectures, how to select and evaluate convolutional neural network models has become an issue that must be considered. Furthermore, regarding model application, direct deployment of deep learning models on mobile terminals requires model pruning and compression, which affects accuracy while also increasing installation package size, making model upgrades and maintenance difficult. To address the aforementioned issues, this study selected AlexNet, GoogLeNet, ResNet, and DenseNet pretrained models for comparative experimental research based on the characteristics of underwater fish real-time recognition tasks. Data augmentation was performed by applying random flipping, rotation, and color jittering to images from the Ground-Truth public fish dataset. Label smoothing was employed as the loss function to alleviate model overfitting, and the Ranger optimizer and Cosine learning rate decay strategy were investigated to further improve model training effectiveness. The precision and recall of each recognition model on the training and validation sets were recorded, and model recognition performance was finally quantified by integrating precision and recall. Experimental results demonstrate that the fish recognition model trained based on DenseNet achieved the highest comprehensive score, with precision and recall on the validation set reaching 99.21% and 96.77%, respectively, and an overall F1 score of 0.9742, indicating that the model' s theoretical recognition accuracy meets expectations. A remote real-time underwater fish recognition system was developed and deployed based on Python. The model was deployed on a remote server, and mobile terminals invoked the fish recognition model through network requests. Actual testing with validation set images shows that under good network conditions, mobile terminals can accurately recognize and display fish

information within 1 s.

Full Text

Underwater Fish Species Identification Model and Real-Time Identification System

LI Shaobo[1,2,3], YANG Ling[1,2,3], YU Huihui[4], CHEN Yingyi[1,2,3]*

[1]College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China
[2]National Innovation Center for Digital Fishery, China Agricultural University, Beijing 100083, China
[3]Beijing Engineering and Technology Research Centre for the Internet of Things in Agriculture, Beijing 100083, China
[4]School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China

Abstract: Rapid and accurate fish identification systems require robust recognition models and effective deployment infrastructure. In recent years, convolutional neural networks have achieved remarkable success in image recognition, yet different architectures possess distinct advantages and limitations. Consequently, model selection and evaluation have become critical considerations. Furthermore, direct deployment of deep learning models on mobile terminals necessitates pruning and compression, which compromises accuracy while increasing package size, hindering model maintenance and upgrades. To address these challenges, this study selected several pre-trained models for comparative experiments based on the characteristics of real-time underwater fish identification tasks. Data augmentation was performed on the Fish Recognition Ground-Truth dataset through random flipping, rotation, and color jittering. Label smoothing was employed as the loss function to mitigate overfitting, while the Ranger optimizer and cosine learning rate decay strategy were investigated to enhance training efficacy. The precision and recall of each model were systematically recorded on both training and validation sets, and a composite metric integrating these measures was used to quantify model performance. Experimental results demonstrated that the DenseNet-based fish identification model achieved the highest composite score, attaining 99.21% precision and 96.77% recall on the validation set, with an overall F1-score of 0.9742—meeting theoretical accuracy expectations. A remote underwater fish real-time identification system was subsequently developed and deployed using Python, with the model hosted on a remote server. Mobile terminals access the model through network requests, and validation tests confirmed that under good network conditions, the mobile application can accurately identify and display fish information within one second.

Keywords: fish identification model; convolutional neural network; model evaluation; Android; Ground-Truth; real-time identification system

1 Introduction

Fish represent one of the most important aquatic products, with over 28,000 known species worldwide. Developing rapid and accurate automated fish identification systems is crucial for fish knowledge dissemination, mixed aquaculture, marine monitoring, and related applications. A complete automated fish identification system must encompass both recognition methodology and model deployment.

Traditional fish identification methods rely on hand-crafted features such as texture, color, and shape. While techniques like Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) have yielded notable improvements, these features require expert manual design and must be re-engineered for different fish species. Such approaches lack generality and suffer from limited accuracy.

In recent years, Convolutional Neural Networks (CNNs) have made tremendous progress in image recognition. Krizhevsky et al.' s AlexNet achieved a breakthrough in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012, winning the competition. Subsequent research has produced numerous improved CNN architectures, including VGGNet, GoogLeNet, ResNet, and DenseNet, all demonstrating excellent performance in image classification tasks. CNN-based algorithms have become one of the best solutions for image recognition problems. However, different CNN models exhibit distinct characteristics, and no single architecture dominates across all scenarios. For instance, VGGNet can learn more features than AlexNet but substantially increases model size, while GoogLeNet' s Inception module significantly reduces parameters but does not address potential degradation issues with increasing depth. ResNet mitigates gradient vanishing in deep networks but introduces implementation complexity and numerous manually-tuned hyperparameters. For underwater fish identification tasks, selecting and evaluating appropriate CNN structures becomes a critical consideration.

Regarding deployment, direct model implementation on Android mobile terminals represents a viable approach. With mobile internet development and smart device proliferation, mobile usage has become ubiquitous. According to the China Internet Network Development Statistics Report, by December 2020, China' s mobile internet users reached 986 million, accounting for 99.7% of all internet users. In 2021, Android systems captured nearly 90% of the mobile market share, making Android-based fish identification highly accessible and user-friendly. However, mobile devices have limited computational capacity compared to servers, requiring deep models to be pruned and compressed before deployment—processes that degrade accuracy and significantly increase application package size, complicating maintenance and upgrades.

To address these issues, this study aims to develop a high-precision fish identifi-

cation system by: (1) investigating multiple mainstream CNN architectures to select and evaluate models suited for underwater fish tasks through comparative experiments, and (2) designing a remote fish identification system to achieve a complete high-precision online solution.

---

2 System Design and Model Construction

2.1 System Architecture

The overall fish identification system design is illustrated in [Figure 1: see original paper]. Users capture fish images through an Android smartphone camera, upload them via network to a server for recognition, which then queries a fish information database to retrieve species profiles, distribution, habits, and life cycle data before returning this information to the Android client for display.

2.2 Data Processing

The Ground-Truth public dataset was used for model training. This dataset contains 27,370 underwater images of 23 fish species manually labeled by marine biologists. [Figure 2: see original paper] displays sample images and quantities for each species. The dataset was split into 80% training data (21,819 images) and 20% validation data (5,474 images). During training, random horizontal flipping, rotation, and color jittering were applied for data augmentation. Images were resized to 260$\times$260 pixels as model input.

2.3 Theoretical Foundation

2.3.1 Convolutional Neural Network Models Convolutional Neural Networks are hierarchical deep learning architectures widely used in image processing. Their primary characteristics include convolutional and pooling layers that progressively extract high-level abstract information from raw input images through successive operations. The final layer computes prediction errors against ground truth via a loss function, then propagates corrections backward through the network until convergence. The convolution operation formula is:

$$i, j + b_L$$

where L represents the network layer, M_j denotes input feature maps, K_i,j represents convolution kernels, b_j is the bias term, and f($\cdot$) is the activation function. Convolution and max pooling operations are illustrated in [Figure 3: see original paper].

AlexNet, VGGNet, GoogLeNet, ResNet, and DenseNet represent classic CNN architectures. Statistical comparisons reveal their model sizes in .

2.3.2 Confusion Matrix and Evaluation Metrics A confusion matrix visually compares predicted versus actual categories. Column totals represent predicted class counts, while row totals indicate actual class distributions. For binary classification, the matrix structure is shown in [Figure 4: see original paper], where TN (True Negative), FN (False Negative), FP (False Positive), and TP (True Positive) represent correct and incorrect predictions for each class.

From the confusion matrix derive key metrics: - **Precision** = TP / (TP + FP) - **Recall** = TP / (TP + FN) - **Accuracy** = (TP + TN) / (TP + TN + FP + FN)

The F1-score, the harmonic mean of precision and recall, provides a balanced evaluation: - **F1-score** = 2 × (Precision × Recall) / (Precision + Recall)

Higher F1-scores generally indicate better model performance.

2.4 Model Selection and Construction

CNN models require large datasets for optimal performance, with more complex tasks demanding larger datasets. Underwater fish images suffer from low resolution, noise, and limited availability due to challenging acquisition conditions and species-specific habitats. Since large-scale underwater fish datasets are difficult to obtain, transfer learning from pre-trained models like AlexNet and ResNet—trained extensively on massive datasets—effectively reduces data requirements and is suitable for resource-constrained scenarios. This study employs transfer learning based on pre-trained models.

As shown in , VGGNet produces the largest model (138 MB), implying greater computational overhead and slower inference. Given the real-time requirements of this task, AlexNet, GoogLeNet, ResNet, and DenseNet were selected for fine-tuning on underwater fish recognition. To mitigate overfitting, Label Smoothing was adopted as the loss function. The Ranger optimizer, combining RAdam and Lookahead, was chosen for its superior convergence properties. A learning rate of 0.0001 with Cosine decay, batch size of 32, and 100 training epochs were used. Hyperparameter configurations are detailed in .

---

3 Results and Analysis

Models were trained on AlexNet, GoogLeNet, ResNet50, and DenseNet169 using the parameters in , then evaluated on the validation set for accuracy, precision, and recall. Results are presented in .

While accuracy is informative, the Ground-Truth dataset exhibits severe class imbalance—the smallest class (black-saddle pufferfish) contains only 16 images versus 12,112 for the largest (damselfish), a 750-fold difference. In such cases, a model could achieve high accuracy by simply predicting the majority class,

making accuracy alone insufficient for evaluation. GoogLeNet yields the smallest parameter count, while AlexNet has the lowest FLOPs but highest parameter count. DenseNet169 offers balanced parameters and computational cost. ResNet50 achieves the highest validation precision (99.26%) but lower recall than DenseNet169. Since precision and recall typically trade off, the F1-score provides more comprehensive evaluation.

Overall F1-scores were computed and shown in . DenseNet169 achieves the highest F1-score of 0.9742, making it the optimal model for this experiment. Despite GoogLeNet' s smaller size, the server deployment environment provides sufficient computational resources, prioritizing accuracy over efficiency. Therefore, DenseNet169 was selected for deployment.

---

4 System Implementation

4.1 Server Implementation

Development Platform: - OS: Ubuntu 20.04 - Environment: Python 3.8, CUDA 11.2, PyTorch 1.7.0, Anaconda 2.0.3, Django 3.2.4, Gunicorn 20.0.4 - IDE: PyCharm Professional 2019.2 - Database: MySQL

Python was chosen for server development. The system was configured on an Ubuntu machine with NVIDIA GT1650 GPU, installing drivers and creating a Conda environment with Django, PyTorch, NumPy, and Pandas.

Project Setup: A Django project was created in PyCharm with the virtual environment configured as the interpreter. Database connections, security policies, and middleware were configured per Django conventions. A FishInfo model was created for database mapping and Django Admin integration, with URL routing established in urls.py. The server project structure is shown in [Figure 5: see original paper].

Model Integration: Following PyTorch documentation, model weights were packaged as *.tar files. A classifier class was created to initialize the model in its `__{{{init}}_{}}` method using PyTorch APIs, with a `predict` function implementing inference logic. This class was imported into Django' s views.py to handle image data and return predictions. Model initialization code is shown in [Figure 6: see original paper].

Fish Information Storage: MySQL stores structured fish information in the FishInfo table, containing species profiles, distribution, habits, and life cycle data collected from marine databases. The table schema is detailed in .

Service Deployment: Django' s built-in development server offers limited performance. For higher concurrency, Gunicorn was used to deploy the service in multi-threaded mode. The server environment and API endpoints are shown in [Figure 7: see original paper].

4.2 Android APP Implementation

Development Platform: - OS: macOS 11.2 - Environment: JDK 1.8.0_{211}, Android SDK Platform 30 - IDE: Android Studio 4.1.2

JDK and Android SDK were installed on macOS with environment variables configured, and development was conducted in Android Studio.

Project Setup: To improve efficiency, the OKHttp lightweight networking framework was integrated for asynchronous request handling. EventBus was incorporated for cross-thread event passing, as Android prohibits UI updates from background threads. Both dependencies are open-source and were added via build.gradle. The APP project structure is shown in [Figure 8: see original paper].

Image Capture and Upload: For maximum compatibility, the APP locks screen orientation to portrait. Android' s Camera API captures image frames in NV21 format, which are converted to Bitmap, compressed to 640$×$480 pixels, encoded as Base64 strings, and uploaded to the server via OKHttp in a background thread.

User Interface Implementation: When server-returned confidence exceeds a threshold, the APP displays fish information. Recognition results are received in a background thread, distributed via EventBus, and rendered on the main UI thread. The APP interface is shown in [Figure 9: see original paper].

4.3 System Testing

Test Environment: - Software: Fish Online Recognition System V1.0 - Mobile: Redmi K40 - Server: Lenovo desktop with Intel i7-8700 and NVIDIA GTX1650 - Network: Campus wireless LAN - Lighting: Laboratory 40W fluorescent lighting

Test Method: Timing code was inserted to measure model inference latency and total round-trip time from request to response. As shown in [Figure 10: see original paper], the APP was installed on the Redmi K40, validation set images were displayed on a monitor, and the phone' s rear camera captured images for real-time recognition, with results and timing displayed on-screen.

Test Results: Testing under these conditions yielded the results in . The APP achieves accurate fish recognition within hundreds of milliseconds, with network transmission being the primary latency component—satisfying real-time application requirements.

---

5 Conclusion and Outlook

This study addresses underwater fish real-time identification by providing a comprehensive framework for model selection, training, and evaluation, and proposes a remote recognition solution for Android mobile terminals with good

extensibility for other image recognition tasks. The system is applicable to fish education, aquaculture, and other scenarios requiring fish identification.

Key findings: (1) The DenseNet169-based model achieved the highest overall F1-score of 0.9742, representing the optimal model. (2) The proposed remote recognition solution demonstrates excellent performance and real-time capability.

Future work should explore model compression and pruning techniques to further improve inference efficiency while maintaining accuracy.

References

[1] NELSON J S, GRANDE T C, WILSON M V. *Fishes of the World*. New Jersey: John Wiley & Sons, 2016.

[2] YANG L, LIU Y, YU H, et al. Computer vision models in intelligent aquaculture with emphasis on fish detection and behavior analysis: A review[J]. *Archives of Computational Methods in Engineering*, 2021, 28(4): 2785-2816.

[3] LOWE D G. Distinctive image features from scale-invariant keypoints[J]. *International Journal of Computer Vision*, 2004, 60(2): 91-110.

[4] DALAL N, TRIGGS B. Histograms of oriented gradients for human detection[C]//2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR' 05). Piscataway, New York, USA: IEEE, 2005: 886-893.

[5] LU H, ZHANG Q. Applications of deep convolutional neural network in computer vision[J]. *Data Acquisition and Processing*, 2016, 31(1): 1-17.

[6] AL-SAFFAR A A M, TAO H, TALAB M A. Review of deep convolution neural network in image classification[C]//2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET). Piscataway, New York, USA: IEEE, 2017: 26-31.

[7] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[J]. *Advances in Neural Information Processing Systems*, 2012, 25: 1097-1105.

[8] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[J/OL]. arXiv:1409.1556, 2014.

[9] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[C]//The IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, New York, USA: IEEE, 2015: 1-9.

[10] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition[C]//The IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, New York, USA: IEEE, 2016: 770-778.

[11] HUANG G, LIU Z, VAN DER MAATEN L, et al. Densely connected convolutional networks[C]//The IEEE Conference on Computer Vision and Pattern Recognition. Piscataway, New York, USA: IEEE, 2017: 4700-4708.

[12] HU H, YANG Y. A combined GLQP and DBN-DRF for face recognition in unconstrained environments[C]//2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017). Paris: Atlantis Press, 2017: 553-557.

[13] NOVAKOVIĆ J D, VELJOVIĆ A, ILIĆ S S, et al. Evaluation of classification models in machine learning[J]. *Theory and Applications of Mathematics & Computer Science*, 2017, 7(1): 39-46.

[14] ERTOSUN M G, RUBIN D L. Probabilistic visual search for masses within mammography images using deep learning[C]//2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM). Piscataway, New York, USA: IEEE, 2015: 1310-1315.

[15] LIN C, LIN C, WANG S, et al. Multiple convolutional neural networks fusion using improved fuzzy integral for facial emotion recognition[J]. *Applied Sciences*, 2019, 9(13): 2593.

[16] BHATIA G S, AHUJA P, CHAUDHARI D, et al. Farm-guide-one-stop solution to farmers[J/OL]. *Asian Journal for Convergence in Technology*, 2019, 4(1). [2021-09-06]. https://asianssr.org/index.php/ajct/article/view/789.

[17] China Internet Network Information Center. China Internet Network Development Statistics Report[EB/OL]. [2021-09-07]. http://www.cac.gov.cn/2021-02/03/c_{1613923423079314}.htm.

[18] SUN Y. 2021 China Mobile Operating System Industry Research Report[R]. Nanjing: LeadLeo Research Institute, 2021.

[19] BOOM B J, HUANG P X, HE J, et al. Supporting ground-truth annotation of image datasets using clustering[C]//The 21st International Conference on Pattern Recognition (ICPR2012). Piscataway, New York, USA: IEEE, 2012: 1542-1545.

[20] MAKANTASIS K, KARANTZALOS K, DOULAMIS A, et al. Deep supervised learning for hyperspectral data classification through convolutional neural networks[C]//2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). Piscataway, New York, USA: IEEE, 2015: 4959-4962.

[21] WEI X. *Parsing Deep Learning: Principles and Visual Practice of Convolutional Neural Networks*[M]. Beijing: Publishing House of Electronics Industry, 2018: 13-14.

[22] RUMELHART D E, HINTON G E, WILLIAMS R J. Learning representations by back-propagating errors[J]. *Nature*, 1986, 323(6088): 533-536.

[23] BOUVRIE J. Notes on convolutional neural networks[J/OL]. (2006-11-22) [2021-09-06]. https://www.researchgate.net/publication/28765140_{{{Notes}}{{on}}}{{{Convolutional}}{{{

[24] KHAN A, SOHAIL A, ZAHOORA U, et al. A survey of the recent architectures of deep convolutional neural networks[J]. *Artificial Intelligence Review*, 2020, 53(8): 5455-5516.

[25] RAHMAD F, SURYANTO Y, RAMLI K. Performance comparison of anti-spam technology using confusion matrix classification[C]//IOP Conference Series: Materials Science and Engineering. Bandung, Indonesia: IOP Publishing, 2020: 12076.

[26] WANG H, ZHAO T, LI L C, et al. A hybrid CNN feature model for pulmonary nodule malignancy risk differentiation[J]. *Journal of X-ray Science and Technology*, 2018, 26(2): 171-187.

[27] KAMILARIS A, PRENAFETA-BOLDÚ F X. A review of the use of convolutional neural networks in agriculture[J]. *The Journal of Agricultural Science*, 2018, 156(3): 312-322.

[28] LIANG H, JIN L, YANG C. Research on underwater target recognition based on deep learning with small sample[J]. *Journal of Wuhan University of Technology (Transportation Science & Engineering)*, 2019, 43(1): 6-10.

[29] ZHANG C, JIANG P, HOU Q, et al. Delving deep into label smoothing[J]. *IEEE Transactions on Image Processing*, 2021, 30: 5984-5996.

[30] GitHub-lessw2020/Ranger-Deep-Learning-Optimizer: Ranger—A synergistic optimizer using RAdam (Rectified Adam), Gradient Centralization and LookAhead in one codebase[EB/OL]. [2021-08-30]. https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer.

[31] LIU L, JIANG H, HE P, et al. On the variance of the adaptive learning rate and beyond[J/OL]. arXiv:1908.03265, 2019.

[32] ZHANG M R, LUCAS J, HINTON G, et al. Lookahead optimizer: k steps forward, 1 step back[J/OL]. arXiv:1907.08610, 2019.

[33] Django overview[EB/OL]. [2021-07-28]. https://www.djangoproject.com/start/overview/.

[34] Gunicorn-Python WSGI HTTP Server for UNIX[EB/OL]. [2021-07-28]. https://gunicorn.org/.

[35] PyTorch documentation[EB/OL]. [2021-07-28]. https://pytorch.org/docs/stable/index.html.

[36] Chinese Marine Fish Database[EB/OL]. [2021-08-30]. http://sea.fundiving.cn.

[37] Taiwan Fish Database[EB/OL]. [2021-08-30]. https://fishdb.sinica.edu.tw/chi/species.php.

[38] Android developers[EB/OL]. [2021-07-28]. https://developer.android.google.cn/studio.

[39] OkHttp[EB/OL]. [2021-08-30]. https://square.github.io/okhttp/.

[40] GitHub-asaskevich/EventBus: [Go] Lightweight event bus with async compatibility for Go[EB/OL]. [2021-08-30]. https://github.com/asaskevich/EventBus.

Note: Figure translations are in progress. See original paper for figures.