Ship Classification in Remote Sensing Images using FastAI

Chitrak Roungroongsom1 and Orachat Chitsobhuk2
Department of Defence Engineering1 and Department of Computer Engineering2
School of Engineering, King Mongkut’s Institute of Technology Ladkrabang.
Bangkok, Thailand.
gravypunk@hotmail.com1 and orachat.ch@kmitl.ac.th2

Abstract—Specifying ship categories in waterways plays an important role in the field of marine surveillance, especially when classification is performed from satellite images due to the advancement in remote sensing technologies. In this paper, we presented an approach for ship classification of optical remote sensing images. Our approach was based on two aspects, modifying models and applying additional techniques to improve accuracy of classification. Two pretrained models, MobileNetV2 and DenseNet121, were modified in this work and all techniques were implemented using Fastai library. To illustrate the effectiveness of our approach, we compared the accuracy of the modified models to the original one. A public Dataset for Ship Classification in Remote sensing images (DSCR), containing six military ship types and a civilian ship type, was used for evaluation. The results showed that our modified DenseNet121 achieved the best accuracy at 99.52% and also outperformed the benchmark result of ResNet101 reported from the original dataset.

Keywords—convolutional neural network (CNN), fastai, remote sensing image (RSI), ship classification

I. INTRODUCTION

Due to the large number of ships in the sea, an effective monitoring system is required for a variety of objectives in both civilian and military approaches, such as sea traffic control, port management, fisheries control, resource protection, and naval defense operations. Ship classification is an important task of such an application for recognizing vessel types in the maritime domain, and it also draws a great attention in research, especially ship classification from images. Convolutional neural network (CNN), a form of deep learning, has demonstrated a great potential in the field of object classification in recent years since [1] won the ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2012 using their proposed model called AlexNet. In the marine environment, CNN has been used to classify ships using images captured from a variety of sensors, including synthetic aperture radar (SAR) images, satellite-based images, and optical images captured by cameras.

Several researches classified ships from optical images. [2] evaluated ship classification using SVM, bag of features, and CNN by AlexNet, and reported that AlexNet outperformed the others. Later studies focused on how to develop or adjust CNN to efficiently categorize ships. However, CNN requires a large amount of data in order to learn more effectively. The authors adopted two transfer learning and fine tuning approaches, as well as data augmentation, a mechanism to synthetically increase images from existing ones, to reuse pretrained models that have already been trained on a new issue. [3] used CNN for ship classification on MARVEL public dataset [4] with 26 classes of civil ships, and compared the accuracy results among remarkable CNN models and reported the optimal tuning for hyperparameters. [5] also used transfer learning with data augmentation on images selected from MARVEL. [6] adapted the VGG16 [7] model, data augmentation, and fine tuning on their own dataset consisting of four types of ship: aircraft carrier, destroyer, oil tanker, and cruise ship.

Three of the aforementioned approaches have been investigated for ship classification on SAR pictures, however there were still problems due to lack of training data. [8] used fine tuning with VGG16 model on SAR images from COSMO-SkyMed to classify three types of ships: bulk carrier, container, and oil tanker. [9] used transfer learning with data augmentation on SAR images from TerraSAR-X to classify the same three types of merchant ships. Even though SAR is a valuable data source, it is severely impacted by speckle noise when utilized in all-weather, all-time situations, making ship interpretation more complicated than with optical pictures. Due to the availability of data sources such as Google Earth, satellite-based images are another alternative for ship classification. Satellite-based images is another choice for ship classification due to the availability of data sources such as Google Earth. [10] used a CNN to classify four ship types: barge, cargo, container, and tanker ship. [11] utilized CNN with preprocessing for ship detection and classification. It resulted that CNN outperformed SVM and neural network. [12] combined extracted features from a CNN with k-Nearest Neighbor method and trained with fine tuning to classify ship and other scenes. [13] used VGG19 model with transfer learning and adam optimization to classify three ship types: oil tankers, bulk carriers, and container ships. [14] presented a public dataset of remote sensing images called DSCR containing general warships for classification task. The author used remarkable CNN models with transfer learning to train for benchmark results.

According to our reviews, the majority of these studies have worked with civilian ship classification and were hardly found to dedicate to military ships since they are rarely observed in common situations. In this paper, we presented a ship classification based on remote sensing images that is primarily focused on military ships from [14]. To enhance classification accuracy, we modified several pretrained models and used different approaches such as data augmentation, one cycle policy training, transfer learning, fine tuning, and test time augmentation.

The paper is organized as follows. Section 2 introduces the basics of CNN. The details of our proposed technique are presented in Section 3. Section 4 shows the experimental results while section 5 is the conclusions.

II. CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks (CNNs) are a type of deep learning model that is widely employed for image analysis. CNN is generally composed of multiple sequences of layers stacked on top of each other. The input images are transformed into arrays of pixel values, which are then processed in those layers capable of recognizing more sophisticated features in hierarchical manners. The basic blocks and related terms of CNN are described as follows:
**Convolutional layer (Conv):** It is responsible for extracting features from input images by sliding several kinds of filters over the input images and performing convolution operations on spatial regions to produce outputs called feature maps. Local connectivity (the filter weights are applied to only a local area at a time to learn local features) and weight sharing are two key concepts presented in this approach (the same filter weights are applied for every location of the input image and embed in the feature map). For an input size I×I×C₁, we can obtain the output size O×O×C₂ of feature map by

\[
O = \frac{1 - F + 2P}{S} + 1 \quad ; \quad C₂ = K
\]

where F is filter size, K is the number of filters, P is the amount of zero padding, and S is stride.

**Rectified Linear Unit function (ReLU):** ReLu is the most widely used activation function. An activation function is usually implemented after the convolutional layer to determine the output of a particular input. It is an element-wise function that returns outputs the same as an input if the input is positive, otherwise returns zero. ReLU6 is a modified version where it returns outputs the same as an input if the input is positive, otherwise returns zero. ReLU6 is a modified version where it limits the activation to a maximum value of 6. For the input x, the output of ReLU6 is obtained by

\[\text{ReLU6}(x) = \min\left(\max(0, x), 6\right)\]

**Pooling layer (Pool):** It is a down sampling process that is used to reduce the spatial size of the feature map and computation in the network. Maximum and average pooling are commonly used functions, which take the maximum and average values respectively. For an input size I×I×C₁, we can obtain the output size O×O×C₂ of pooled feature map by

\[
O = \frac{1 - P}{S} + 1 \quad ; \quad C₂ = C₁
\]

where P is pool size, S is stride, and C₁ is the number of channels from the previous convolution layer that remains unchanged.

**Fully connected layer (FC):** It takes a flattened input and connects every input neuron to every output neuron in another layer. Linear operations are used to learn the combinations of the high-level features from convolutional layers. We can add one or more FC layers to form a classifier at the end of a CNN.

**Softmax function** is an activation function that turns input values into a normalized probability distribution over output classes. So the output will be varied between the value of 0 and 1 which are suitable for representing the class scores. The formula is as follows

\[
\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}
\]

where x is the input vector, which can be any real value and k is the number of output classes.

**Loss function** is a way to measure the difference between true labels and predicted labels and we aim that difference to be less. We can convert to the minimum of the loss function using the derivative of the loss function with respect to weights and use it to update the weights in backpropagation. Cross entropy loss is the one that widely uses in classification task and the formula is given by

\[
\text{Cross Entropy Loss} = -\sum_{i=1}^{k} y_i \log(\hat{y}_i)
\]

where y is the vector of actual label, \(\hat{y}\) is the vector of classifier output which is actually the output from softmax results, and k is the number of output classes.

**Optimizer** is an algorithm used to tune the parameters of a neural network in order to minimize the loss function. Adaptive Moment Estimation (Adam) [15] is the most often used since it is computationally efficient and has little memory requirements.

For training with a limited dataset, a situation called “overfitting” may occur. It happens when a model learns too much details and noises in the training data and becomes too attuned to them. Finally, the model will lose the ability to generalize and cannot effectively predict on new data. Therefore, additional techniques should be applied to alleviate this problem, such as, batch normalization, dropout, and data augmentation.

**Batch normalization (BN):** [16] It is a technique to standardize the inputs to a network for every mini-batch. Mini-batch refers to one batch of data supplied for any given epoch, a subset of the whole training data. Let x be a mini-batch of activations of the layer to normalize. To normalize x, we replace it with

\[
y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \ast \gamma + \beta
\]

where y is the output of normalized x, \(\mu\) is the mini-batch mean, \(\sigma^2\) is mini-batch variance. \(\epsilon\) is a small number added to variance to avoid dividing by zero. \(\gamma, \beta\) are learnable parameters for scaling and shifting of the normalized value.

**Dropout:** [17] It is a technique that some nodes are randomly ignored during training. Each node comes with the probability that not to be fired in forward propagation and not to be updated in backward propagation.

**Data augmentation:** It is a technique used to increase the number of images by creating multiple copies of modified images from the existing ones. Various kinds of processing can be applied for modification, for example, flipping, brightness adjustment, cropping, rotation, and so on.

**Transfer learning and fine tuning:** One thing to remember about deep learning model is that it requires a large amount of data and also computing resources to train the network. Fortunately, there are two efficient methods for dealing with this problems: transfer learning and fine tuning. Pretrained models, which are state-of-the-art models that have been trained on a large public datasets such as ImageNet, are used in both techniques. The transfer learning leads us to the optimal weights that comply with varieties of basic classes.
from millions of images and are ready to be used without training from scratch and we can reuse it for our specific task. The thing that differentiate those two methods is how we train it. For transfer learning, we usually freeze the layers of feature extractor part and train the classifier part on the assumption that low level layers share common features, such as corner and edges. This means that we preserve the weights from the feature extractor the same and submit them to the classifier for training. For fine tuning, an approach of transfer learning, we unfreeze and retrain the feature extractor part of the model to update the weights for learning data in backpropagation.

One cycle policy: [18] Cyclical learning rates (CLR) is a method for setting learning rate to oscillate between the specified minimum (min_lr) and maximum (max_lr) bounds. The 1cycle policy suggests to use CLR as only one cycle, which consists of two steps, firstly, increasing the learning rate from min_lr to max_lr and secondly, decreasing it vice versa. After the cycle is completed, the learning rate will be decreased slightly less than its initial min_lr. For the momentum, similar concept is applied but in reverse steps. Using this method, networks can be trained much faster than using the standard training methods.

Testing time augmentation (TTA) [19] is the way we applied data augmentation to the test set. Four augmented images are created from the original one. All five images are predicted and the results will be averaged to get the final prediction.

III. THE PROPOSED APPROACH

Our ship classification was implemented using fastai. [20] Fastai is a deep learning library based on PyTorch framework that is easy to use and achieves good performances. It provides bunches of built-in functionalities, such as, data augmentation, and state-of-the-art models allowed us to create an image classification application with transfer learning or fine tuning. In this study, we chose two outstanding pretrained models, MobileNetV2 [21] and DenseNet121 [22], as the base models. The summary of our proposed approach is as follows:

1) For input data, we used data augmentation to increase the number of images.
2) For the proposed model, we added a custom block and changed the classifier part of the base model.
3) For training, we used the 1cycle policy to train. The training was experimented for both transfer learning and fine tuning approaches.
4) For testing, besides the normal predictions, we also showed the prediction results provided by the TTA technique.

A. Data Augmentation

Below is the set of configurations of data augmentation in fastai that we used in our experiments.

- **resize** = 224 (for item_tfms)
- **size** = 128 (for batch_tfms)
- **do_flip** = True (horizontal flip)
- **flip_vert** = True (vertical flip)

- **min_scale** = 0.75
- **max_rotate** = 180.0
- **max_zoom** = 1.2
- **max_lighting** = 0.2
- **max_warp** = 0.2

B. The Proposed Model

We utilized the pretrained model for transfer learning by cutting off the classifier part so the feature extractor part remained the same and we called it base model. Then we connected our additional block to the end of the base model where N is the number of filters from the last layer of base model, N = 1280 for MobileNetV2 and N = 1024 filters for DenseNet121 in this case. Finally, we used the head layers provided by fastai implementation [23] as a classifier. Fig.4 showed the architecture of our proposed model.

C. Training and Testing Configurations

Here are the hyperparameters and related configurations in training and testing.

- **Batch size** = 32
- **Number of epochs** = 10 – 40
- **Learning rates:**
  - Transfer learning = 4e-5 (min at start), 1e-3 (max)
  - Fine tuning = 2e-4 (min at start), 1e-3 (max)
- **Momentums** = 0.85 (min), 0.95 (max)
- **Loss** = Cross Entropy Loss
- **Optimizer** = Adam
- **Metric of evaluation** = Accuracy

Note that for the learning rates and momentums, we just left them as the default values provided by fastai, and they would be varied by cosine annealing according to the 1cycle policy. Fig. 5 showed examples of learning rate and momentum plots over iterations of 10 epochs. For TTA, the set of augmentations would be randomly selected to apply the same way as mentioned in 3A.
IV. EXPERIMENTS AND RESULTS

A. Dataset

DSCR public dataset created by [14] was used in this experiment. It contains 20,646 total images separated to 13,961 images for training, 3,354 images for validation, and 3,331 images for testing. There were seven categories of ship in the dataset: aircraft carrier, assault ship, combat ship, cruiser, destroyer, civilian ship, and other military ship. The author collected images from three public datasets and Google Earth from 29 ports all around the world in 2000 – 2018 periods. The images were in range of resolutions around 0.5 - 2 meters. Fig. 6 and 7 showed the number of images and sample images from DSCR dataset.

B. Training Schemes

We implemented the proposed approach using fastai v.2.2.7 on Jupyter Notebook in Windows 10. NVIDIA GeForce GTX 980 GPU with 4GB GDDR5 and CPU Intel Core i7-6700HQ processor with 32GB of RAM were used for training. We divided the training experiments to main five plans for each base models. The description of the plans appeared in Table I is shown as follows:

1) For the first plan, we referred as “Base model” architecture with 10 epochs of transfer learning as the baseline plan. This plan is the way we used the base model without custom block and trained without one cycle policy (learning rate at 0.001).

2) For the second plan, we referred as “Base model + Add block + Head” as our proposed model and used this architecture for the rest of experimental plans. Only the ways we trained were different. This plan was accomplished with 10 epochs for training of transfer learning. It means that we froze the base model part, then trained the additional block and head (classifier) part with one cycle policy.

3) For the third plan, we unfroze the whole proposed model to train for 10 epochs using fine tuning approach.

4) For the fourth plan, we simply trained for both transfer learning and fine-tuning approach. We first trained the proposed model using transfer learning for 10 epochs similar to the second plan, then we further trained using fine tuning similar to the third plan for the next 10 epochs.

5) The last plan is similar to the forth plan except we increased the number of epochs to 20 for both approaches.

The scheme (1) – (5) are the plans we applied for MobileNetV2 as the base model and the scheme (6) – (10) for DenseNet121. Note that fine tuning implemented by fastai will first train one epoch of transfer learning before unfreezing the layers. Thus, the number of epochs that used for fine tuning will be plus an extra epoch.

C. Experimental Results

Table I showed the training results. Let’s look at the first two schemes, (1) and (2), which used MobileNetV2 as base model and trained with the same 10 epochs of transfer learning approach. The scheme (1), the baseline plan, used original base model and trained with fixed learning rate while the scheme (2) used the proposed model (altering the base model with our custom part) and trained with one cycle policy. We obtained that the scheme (2) provided higher accuracy than the baseline scheme (1). The scheme (6) and (7) of DenseNet121 as the base model also illustrated the same result as the scheme (7), only a little higher accuracy than the scheme (6). Even though adding more hidden layers will help the
model extract more features leading to the higher accuracy results, the overfitting problem could occur in some cases. Our approach handled this problem by including batch normalization and dropout layers in the proposed model, and also increasing data using augmentation. Those techniques helped the model to reduce the impact of overfitting as it has been seen from Fig. 8 and 9. Fig. 8 showed the training and validation losses over iterations from the scheme (5) while Fig. 9 illustrated results from the scheme (10). Both training losses (blue lines) and validation losses (orange lines) are moderately decreased and close to each other at the end. This implied that the models were well enough to predict the validation sets.

Moreover, we observed that using the proposed approach, we can achieve good accuracy even in the first few epochs due to the one cycle policy of training. We can train more to achieve the acceptable results since the accuracy has naturally grown as we increased the number of epochs.

Fig. 11 displayed the confusion matrix of the scheme (5) which provided the best result for MobileNetV2. The scheme (10) was the best result for DenseNet121 as shown below in Fig. 12.

According to the plot in Fig. 10, DenseNet121 started with the higher accuracy than MobileNetV2. The reason is that MobileNetV2 is designed as a lightweight model with less depth and parameters compared to DenseNet121. Thereby DenseNet121 can learn features better according to its characteristics. Although MobileNetV2 is inferior to DenseNet121 by default, the proposed approach allowed the model to improve learning and delivered higher accuracy than the baseline model. Even though the accuracy result of the proposed MobileNetV2 was slightly less than DenseNet121, it still provided the preferable results with less training time.

V. CONCLUSIONS

In this work, we proposed an approach for ship classification on remote sensing images. The dataset we used was from the DSCR containing seven categories mostly in military ships. Our proposed approach was the method to customize the existing deep convolutional neural network model and apply various techniques including, data augmentation, one cycle policy training, transfer learning, fine tuning, and test time augmentation to improve the accuracy of classification. The application was built using fastai library.
We demonstrated the performance of the proposed approach on two outstanding models and found that the proposed DenseNet121 achieved the best accuracy at 99.52%. It is not only superior to the baseline model but also the reported benchmark result from the DSCR dataset.

For the future work, this proposed approach can be extended further on fine-grained military ships since the dataset in this work was classified in coarse grained categories or other types of images such as SAR and optical camera images. For image classification task, one image is assumed to belong to one class of the defined scope. Therefore, it can be extended the study on object detection in the case of multiple ships or other objects in image.

ACKNOWLEDGMENT
This research was supported by The Defence Technology Institute, The Kingdom of Thailand; Department of Defence Engineering and Department of Computer Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

REFERENCES