CS7643 Project Report: Deep Learning for Ship Detection and Classification

Kalpan Dharamshi
kdharamshi3@gatech.edu

Ty Martz
ty.martz@gatech.edu

Valerie Schnapp
vschnapp@gatech.edu

Abstract

Ship detection and classification would be influential for naval security as well as different marine traffic organizations. The current standard would be an actual human in a light house or lookout post detecting and classifying ships coming to port. Our goal in this paper is to improve these undertakings using deep learning (DL) in an artificial intelligence (AI) automated process. In an attempt to classify and detect ships in a large set of data, the team ran a case study in which various models were evaluated and fine-tuned for this issue. The best performing model architecture was chosen for both classification and object detection. The team used the ImageNet pre-trained model available on the PyTorch Zoo, and fine-tuned the last few fully connected layers. The classification model performed with a test accuracy of approximately 85%. This kind of model could be used along with modern hardware such as drones and advanced imaging technology to have real-time knowledge of marine traffic.

1. Introduction/Background/Motivation

Being able to identify incoming ships quickly would be a major benefit to organizations that deal in marine traffic such as the Coast Guard, the United States Department of Defense (DoD), or even a local marina. The different types of ships to classify are cargo, military, carrier, cruise, and tanker ships. Knowing that a ship is coming up the horizon can mean several things: a cargo ship is coming in with consumer imports, a cruise ship is docking full of citizens, or even other countries are coming over to attack by military boat, meaning that the Navy needs to be prepared. All ships have similar structures but it is the small details in which they differ. Military ships have a superstructure in the middle while a cargo ships are stacked with countless containers. A motive of the model was that it must be able to pick up on and identify key features within each class for it to achieve a desirable outcome.

Another use case for this project could be detecting and tracking ships at sea using satellite imagery. Although satellite imagery is out of the scope of the project, this would be helpful to the same parties listed above who don't just want to know what is coming up the horizon, but what is on track for delivery. Today, relying on communication tools or the eyesight of humans is prone to error and not ideal. This current practice is outdated given technology today that can be used to identify and communicate movement of ships in an automated and reliable manner. Although the team did not explore satellite images in this case study, experimentation on object detection was performed.

When looking at the current state-of-the art practices for image classification and object detection, there are a number of stand out models. In the image classification realm, the CoCa (Contrastive Captioner) model has one of the best accuracy scores on the ImageNet data set with a top 1 accuracy of 91%. This model leverages and encoder-decoder framework jointly with contrastive loss and captioning loss. For the object detection problem, a top performing model is the DINO (DETR with Improved DeNoising) model. This architecture includes convolution layers with a transformer and improves on prior models through denoising training and improving initialization. DINO has one of the best Average Precision scores when tested on the COCO data set and is considered state-of-the-art.

A handful of DL models were compiled, trained, and tested. The pre-trained image models used for experimentation on the classification problem include ResNet152, Alexnet, ConvNeXT, and Swin. As for object detection, the focus stayed on the Faster RCNN model with DeepLabV3 for segmentation.

2. Approach

2.1. Data

The team used data downloaded from Kaggle [2] with nearly 6,000 different images in JPG format. The images came with labels ranging from 1 to 5 and required a mapping to each of the five classes of ships for the model to identify. Although there are many different ships styles, only these 5 specific ships with are used for this study as they were already labeled. Unfortunately there were some inconsistencies as there were images with varying sizes and channels as well as some images in grey scale as opposed to RGB. In the training set of approximately 5,000 images,

there were 151 unique shapes of the images giving us an initial consideration when working with this data (Figure 1).

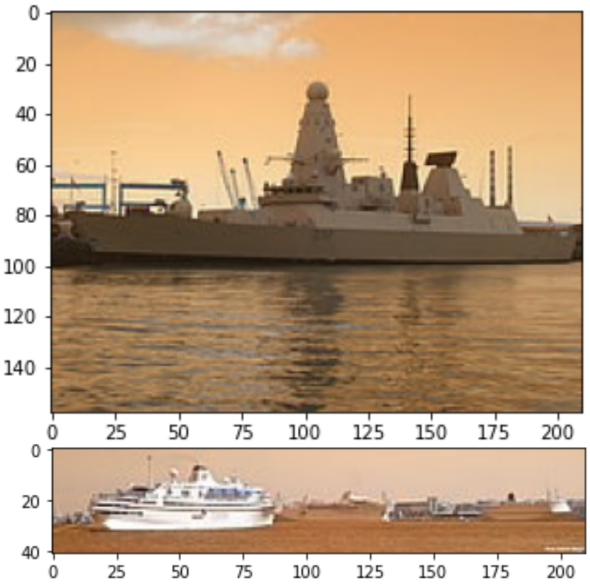

Figure 1. Images of varying sizes

Along with variety of size and shape, there was also a variety in quality and noise, making for a well-rounded data set for this use case. Some images had clean views of the object in question while others included much more noise. Using edge detection shows the extent of activity going on around the ship (Figure 2). These quirks of the data were kept in mind during analysis, processing, model-building, and testing. Given all of these early insights on the data, the team had to improve the standardization of the images and prepare it for model tuning and testing.

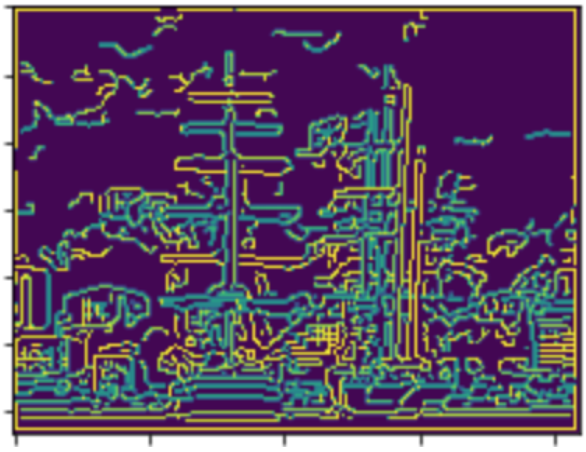

Figure 2. Some images include a good deal of noise

2.2. Pre-Processing

Pre-processing was introduced to standardize the image sizes and channels, as well as feed in a consistent image tensor to the ConvNeXT model. The standardization process included a simple procedure to pad lower dimensional images with zero pixel values.

Recall that the five different types of ships being classified are: cargo, tanker, military, carrier, and cruise. To keep consistency the label range was updated from 1-5 to 0-4 by replacing the fives with zeros. The data was fairly balanced among classes, so no special processing was needed to balance the classes before training (Figure 3).

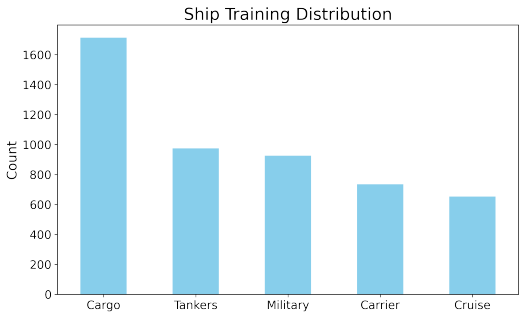

Figure 3. Distribution of ship categories in the training set

Although the data had already been defined as train or test images from the Kaggle download, the testing set did not come with labels to check on. As a result, the team randomly, but distributively, selected images to pull from training set with an already defined classification. Therefore, when the testing was actually being done on images the model had never seen, it would be known whether or not the model predicted accurately.

2.3. Data Platform

It should also be noted that to run the models, the team launched a Deep Learning Virtual Machine (VM) inside the Google Cloud Platform (GCP). This instance used one GPU from Nvidia with the PyTorch platform pre-installed for us. Using this VM was essential for the classification program as it trained the model in a few hours on CUDA, opposed to days using just CPU. Members of the team did not have to spend money out of pocket as the education coupons took care of the cost.

2.4. Transfer Learning

The team utilized a transfer learning approach to initialize the model weights for the classification use case. The ConvNeXT model has been initialized with ImageNet weights and the final fully connected layer of the model has

been replaced with the problem specific layer to perform appropriate training and classification. The entire ConvNeXT model architecture has been frozen, and only final layer is trained for classification fine-tuning.

2.5. Success and Novelty

The team evaluated most of the available models in the PyTorch Zoo and studied recommended architecture to build and train the models for classification and object detection. The results obtained after comparisons and studies gave the ample of confidence that the selected approach adheres to the industry standards and recommended architectures for computer vision (CV). However, it is understood that in the future there may be better architectures that can provide even better results than today. DL is an evolving field where newer practices and architectures are established on a regular basis.

2.5.1 Classification

The best performing model for classification made use of the ConvNeXT architecture. This model uses novel activation and normalization approach to provide better results than architectures like ResNet150. It uses the GELU (Gaussian Error Linear Unit) activation functions which is considered to provide better gradient flow than ReLU activation functions. It uses normalization only before 1X1 conv layers and it only uses layer normalization instead of batch normalization.

2.5.2 Object Detection

The team also performed object detection using several approaches. The PyTorch deep learning libraries were heavily relied upon in this process. The OpenCV library was also implemented to verify model results in two different ways. The first was to draw a bounding boxes around the objects detected in the images and label them. The second was to use semantics to separate the background from detected objects. An example of this can be found in Figure 4.

Figure 4. Results of object detection model on a test image

2.6. Problems Encountered and Solutions

The team had anticipated that both classification and object detection would be the challenging experiments. During initial investigation, it was found that there are multiple options available that can classify, detect and segment images. However, not all the frameworks available in the industry use the DL approach and are not reliable in terms of speed and accuracy. The data provided to was useful for the classification part of the project. Combining the OpenCV library with DL models in PyTorchVision ended up being the most successful approach to perform object detection.

The ship data set consisted of images of different sizes and different channels. Most of the images had RGB channels, but some of them were also gray scale. A data preprocessing step was introduced before it could be fed to model for training. The pre-processing step included a simple procedure to standardize all the images to have the same dimensions and channels. To resolve this dimension issue, zero pixel padding was used on the images as well as channels with zero pixels for the gray scale images.

As mentioned earlier, the Kaggle website provided a separate training set and test set for competition, but the provided testing set was not used as it did not contain the category labels. The training set had more than six thousand images available for the model training, so it was decided to split the training data set further into train set (5,000 images) and test set (1,200 images). The modified test data set had category labels associated to it, so team could test the classification accuracy of the model.

The team programmed the classification and detection logic based on the learning's from the course content and assignments. The transfer learning approach and modification of the final fully connected layer was learned from the PyTorch tutorials and discussion forums.

3. Experiments and Results

3.1. Ship classification

Multiple architectures were evaluated and their test accuracy's and loss curves were used to measure their performance. ConvNeXT architecture provided us the most test accuracy and hence it was selected to perform ship classification (see Figure 6). The accuracy scores of the fives classes can be found in Table 1. The Military ships had the most accurate classification of approximately 96% whereas the Tanker ships had the lowest accuracy of approximately 74%. A big reason for the miss-classification is due to the defining features not standing out. In the future, unfreezing further layers of the architectures and tuning them would improve the capability of picking up the more defining features.

Team has used Adam optimizer to search the optimal weights of the neural network. Adam optimizer uses a

Class 0	Class 1	Class 2	Class 3	Class 4
Tanker	Cargo	Military	Carrier	Cruise
0.7366	0.7961	0.9604	0.8564	0.9000

Table 1. Final accuracy of ConvNeXT model over the 5 classes

combination of velocity and gradient descent to take larger strides towards minima. Learning rate and weight decay parameters were used for hyper parameter tuning, and the hyper parameter curves (see Figure 5) indicate that default values provide better test accuracy than any of the tuned parameters.

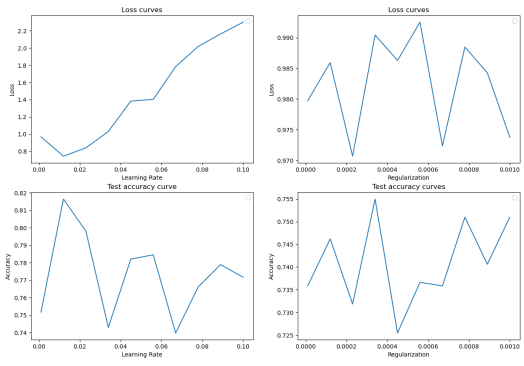

Figure 5. Hyper parameter curves for ConvNeXT

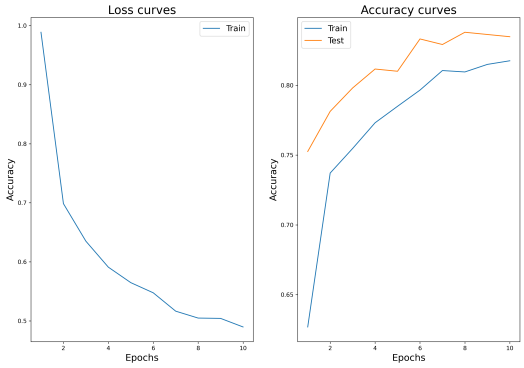

Figure 6. Loss and test accuracy curves for ConvNeXT

The experiments started with reading multiple available architectures in the PyTorch Zoo for classification. Architectures like AlexNet, ResNet150, and SwinTransformer were used for evaluation. The models were loaded with pre trained weights from ImageNet. The final layer of the architecture was modified for ship classification and only it was updated in the backward pass. The rest of architecture was frozen to avoid over fitting, and consistency of evaluation.

The experiments succeeded in providing us a novel architecture to study and evaluate. The ConvNeXT architecture is inspired from the ViT (Vision Transformers) and earlier convolution neural networks architectures like ResNet to build a new approach to understand the images, and optimize the neural network weights. The GELU activation function and the layer normalization approach are the two approaches that help it to gain better performance over Transformers and ResNet models. It uses depth convolution similar to multi head attention mechanism in Transformers to aggregate the global information in the image. The model also utilizes inverted bottleneck to concatenate the image patches it created and propagate it to the deeper layers of the network (see Figure 7).

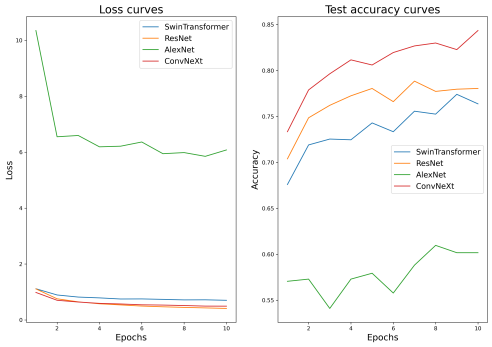

Figure 7. Loss and test accuracy curves of different architectures

3.2. Ship Detection

Several architectures were once again tested for object detection and semantic segmentation. The teams initial instinct was to try the Faster RCNN ResNet50 model. The team was able to measure success by using the OpenCV package where if an object was detected within the specified threshold, a rectangle was put around the object with a label. With a threshold 85%, only 5 out of 10 the 10 test images were successfully detected. This was considered a failure for us as 50% of images did not have detection. The team didn't give up and continued iterating and tuning in hopes for improvement. After changing this to the Faster RCNN ResNet50 V2 TorchVision model, all 10 images were detected on a 90% threshold (see Figure 8). This was considered a success with each sample detected with high confidence.

Further, the team wanted to segment out the images to measure model successes. For image segmentation, the initial model of Mask RCNN ResNet50 only identified two thirds of the test images. Using the DeepLabV3 ResNet50 architecture showed to be more successful when it was able

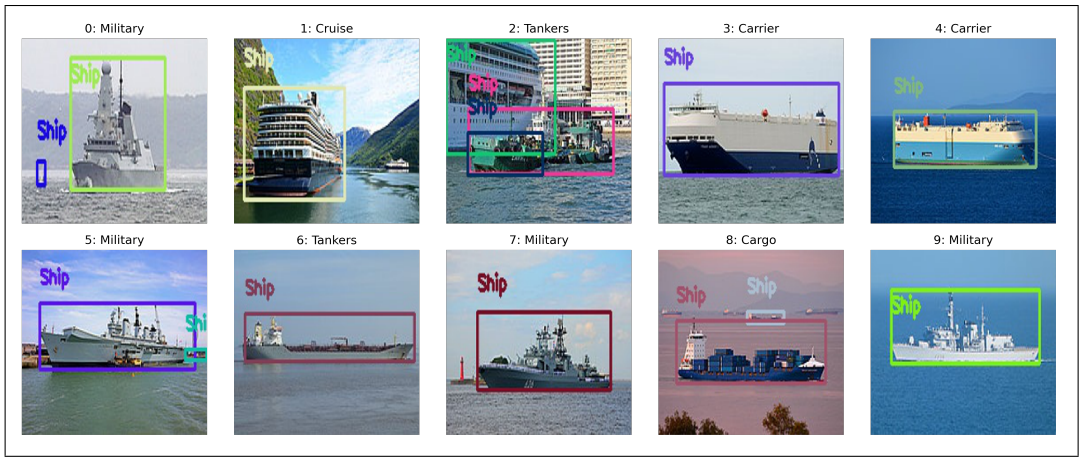

Figure 8. Object Detection on all 10 test images

to show semantic segmentation on all 10 test images (see Figure 9). The OpenCV package was once again used to implemented the semantics from the segmentation detection.

All in all, there was no software that could be used to measure the success and had to use provided knowledge of the images to do so. It was known that all of the test images had at least one ship in it. Therefore, if there was no rectangle drawn over the image it meant that the detection failed. On a larger scale, a different approach would be needed to measure success because when it comes to imagery in the real world, there may not always be a ship in the view.

4. Conclusion and Future Work

The team has adopted an pragmatic approach to solve the ship classification and detection problem. Training deep neural networks from scratch is costly and time consuming affair. In software industry, transfer learning is used to retrieve the base optimal weights required to solve the problem, and fine tuning is performed on the base model to solve the problem at hand. Our approach is simple and effective that can be understood and adopted by anyone.

There are many paths forward for this research whether it is using general models or specifically fine-tuned ones to solve the problem. The future is bright for ship classification and object detection as the expected improvements in software and picture quality improve. It is promising that there are continuous works in general CV models that are robust to numerous classes. Continuing the research would involve a greater distribution of images for training and testing to add more variation in background (eg. ships on land) and camera angle (eg. satellite vs horizon).

Future challenges include changes in ship styles which could change the feature distribution of classes and confuse stale models. Researchers must continue to expand on the models and techniques used while also increasing the breadth of potential ship classes that come through in unique images. For this specific use case focused on marine traffic, a fine-tuned model discovery may be better. The key focus begins with ships but can expand further to marine life and unknown floating objects in bodies of water. All of this research could be used for different institutions to better understand their aquatic territories.

5. Work Division

Kalpan, Ty, and Valerie all contributed to this final project using their different strengths.

Kalpan lead the classification piece, Valerie lead the object detection piece, and Ty lead the analysis piece. However, all three members helped out in all sections of the project.

After submitting the project proposal, their fourth member decided to withdraw from the class. The three remaining members took on the extra work they scoped out and were able to manage and complete it.

Further details of work division and contributions can be found in Table 2.

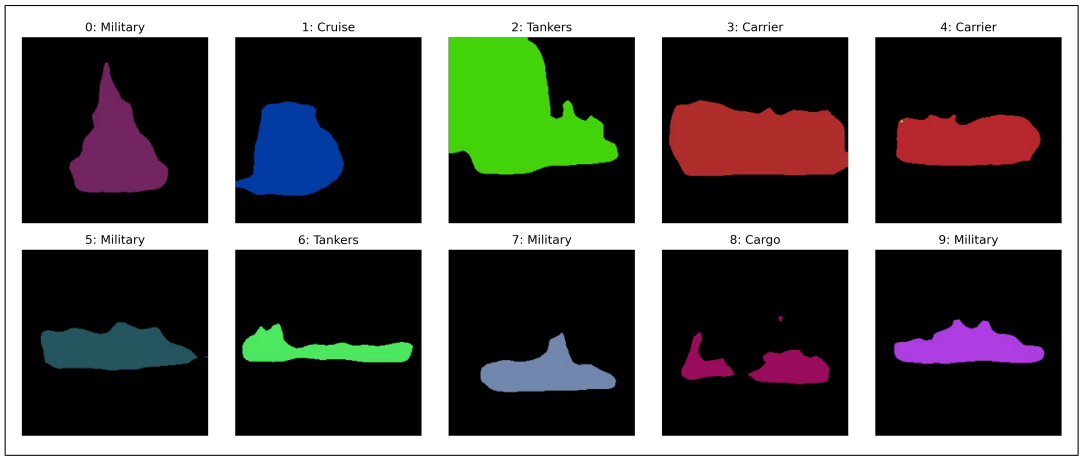

Figure 9. Object Segmentation on all 10 test images

Student Name	Contributed Aspects	Details
Kalpan Dharamshi	Classification Model and Hypertuning of Object Detection Model	Kalpan programmed the classification piece and ran experiments for object classification. He also laid out the groundwork for object detection piece, and wrote section 2 and 3.1 in the report
Ty Martz	Data Exploration and Lead Analysis Writer	Ty performed data analysis to understand the images used throughout the project and created visualizations to accompany them. He also led the charge on the paper writing, flow, and formatting.
Valerie Schnapp	Object Detection Models and Hypertuning of Classification Model	Valerie wrote the code and ran experiments for the object detection and segmentation piece as well as generating the images for them. She also did some hypertuning for the classification model as well as some of the analysis write-up (specifically section 3.2 and the abstract).

Table 2. Contributions of team members.

References

[1] Rob Almeida. Ship identification 101, 2015.

[2] Arpit Jain. Game of deep learning: Ship datasets, 2019. 1

[3] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[4] Daniel Moraite. Detecting ships in satellite imagery, 2019.

[5] Milena Napiorkowska, David Petit, and Paula Marti. Three applications of deep learning algorithms for object detection in satellite imagery. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 4839–4842, 2018.

[6] Abhinav Sagar. Deep learning for ship detection and segmentation, 2019.

[7] Ning Wang, Yuanyuan Wang, and Meng Joo Er. Review on deep learning techniques for marine object recognition: Architectures and algorithms. *Control Engineering Practice*, 118:104458, 2022.

[8] Xue Yang, Hao Sun, Kun Fu, Jirui Yang, Xian Sun, Menglong Yan, and Zhi Guo. Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks. *Remote Sensing*, 10(1):132, jan 2018.

[9] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.

[10] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022.