Project Final Presentation

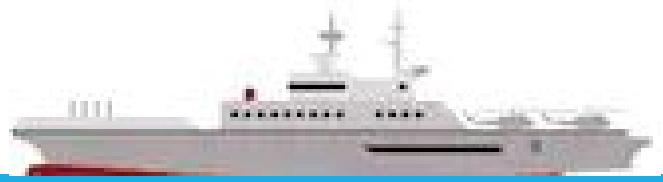
SHIP TYPE CLASSIFICATION USING CNN

Abdülkerim Mustafa DEMİR

514191028

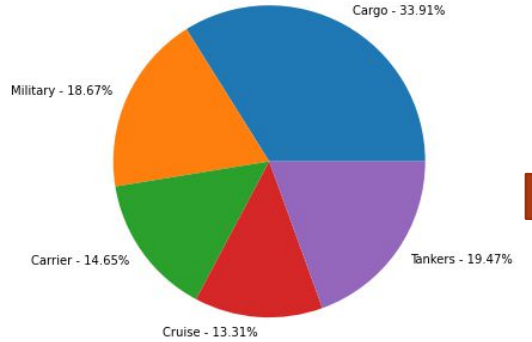
Problem Statement

1. The types and characteristic structures of the ships are different in terms of their intended use.
2. Countries must be aware of maritime traffic in order to follow the research in the exclusive economic zone and to protect their maritime relevance and interests.
3. With this project, it is purposed to create a model that determines ship types using our data set consisting of ships.

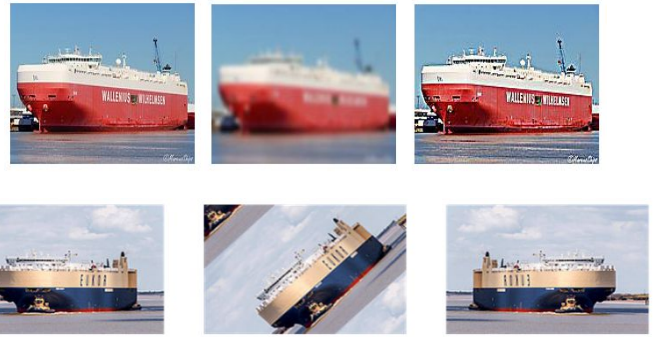


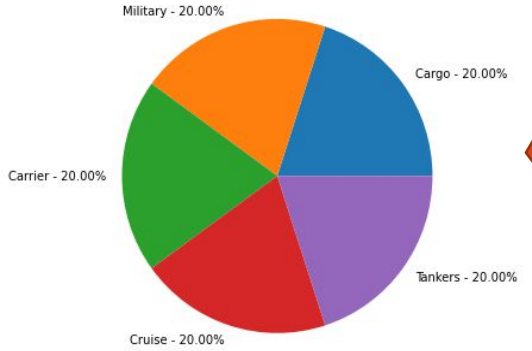
WHAT HAD BEEN DONE?

- **DATASET**



- **DATA AUGMENTATON**
  - Rotation
  - Flipping
  - Mirroring
  - Blurring
  - Unsharp Masking





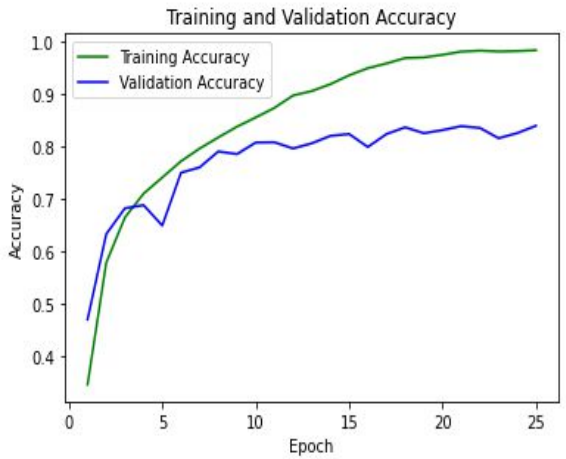
MODEL ARCHITECTURE

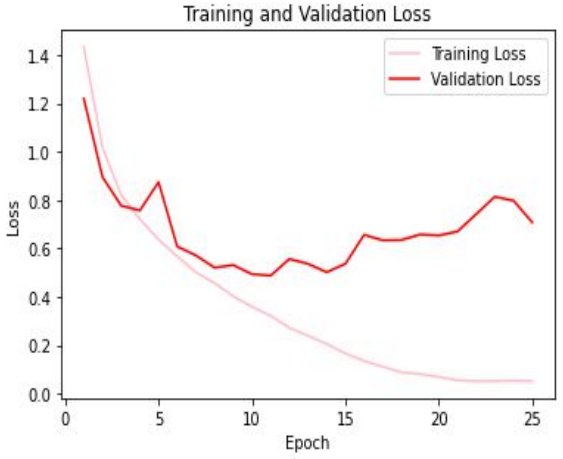
Baseline Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 64)      1792
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 256)       295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 256)       590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 16, 16, 512)       1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 512)       2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 16, 16, 512)       2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 512)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 8, 8, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              33558528
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 20485
=================================================================
Total params: 65,075,013
Trainable params: 65,075,013
Non-trainable params: 0
_________________________________________________________________
```

- **Batch Size** : 32
- **E-pochs** : 25
- **Activation** : ReLU
- **Optimizer** : Adam
- **Learning Rate** : 0.00001

- **Training Accuracy** : 98.79%
- **Validation Accuracy** : 83.99%





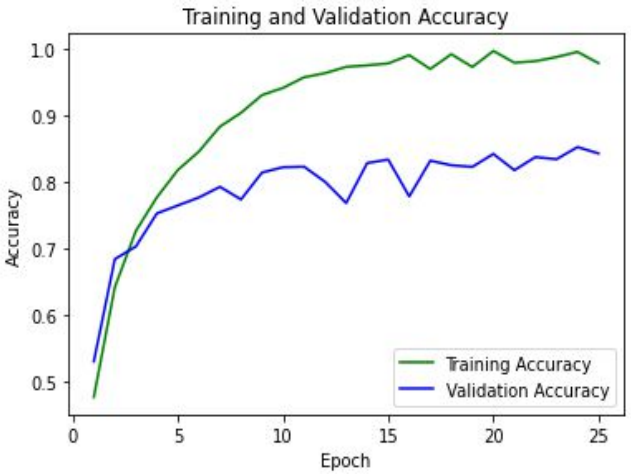
OPTIMIZATION

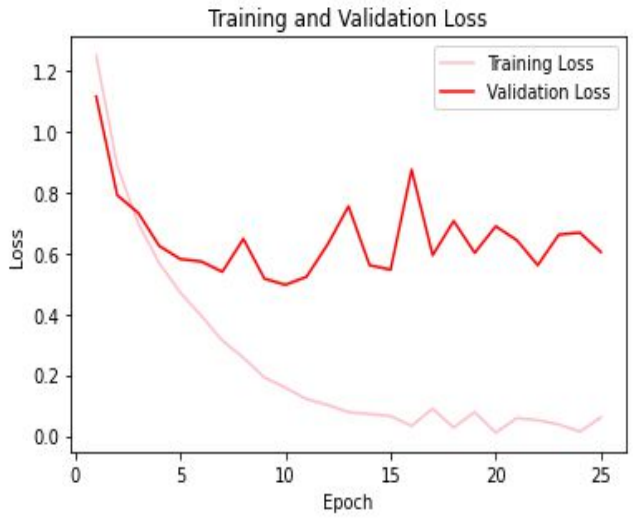
1. INITIALIZER

- He Initialization

• **Batch Size**	: 32
• **E-pochs**	: 25
• **Activation**	: ReLU
• **Optimizer**	: Adam
• **Learning Rate**	: 0.00001

• **Training Accuracy**	: 99.69%
• **Validation Accuracy**	: 85.32%





OPTIMIZATION

2. OPTIMIZER

- One of the most commonly used optimizers to train models in deep learning is Adam. In this project, we used Adam as an optimizer while compiling our
- In order to compare Adam's success, I compiled the model that we created last by choosing SGD as the optimizer models.

- **Training Accuracy** : 75.65%
- **Validation Accuracy** : 70.57%

- We got very low accuracy compared to Adam optimizer

OPTIMIZATION

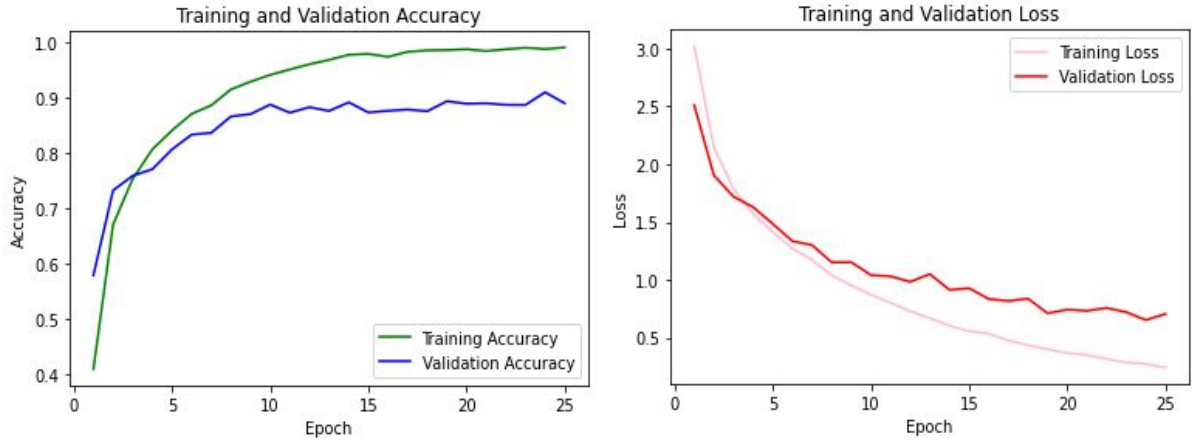
3. REGULARIATION

- **A. Dropout**

	Dropout (Fully Connected Layers)	Training Accuracy	Validation Accuracy
Trial 1	0.5	98.17%	86.25%
Trial 2	0.4	98.87%	85.77%
Trial 3	0.2	99.01%	89.87%

- **B. L2 Regularizaiton**

	L2 Regulazition (Fully Connected Layers)	Training Accuracy	Validation Accuracy
Trial 1	0.001	98.99%	90.10%
Trial 2	0.0001	99.39%	90.89%
Trial 3	0.00001	99.01%	90.00%

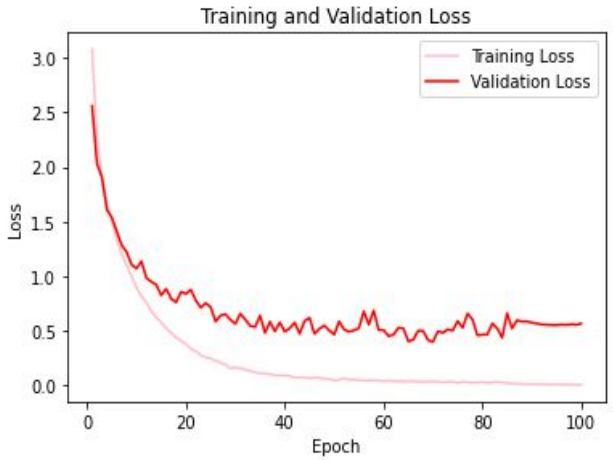


Training & Validation Accuracy – Trainig & Validation Loss
after L2 Regularization (According to 0.0001)

OPTIMIZATION

4. NUMBER OF EPOCHS

	Number of Epochs	Training Accuracy	Validation Accuracy
Trial 1	25	99.39%	90.89%
Trial 2	50	99.67%	90.72%
Trial 3	100	100%	91.60%



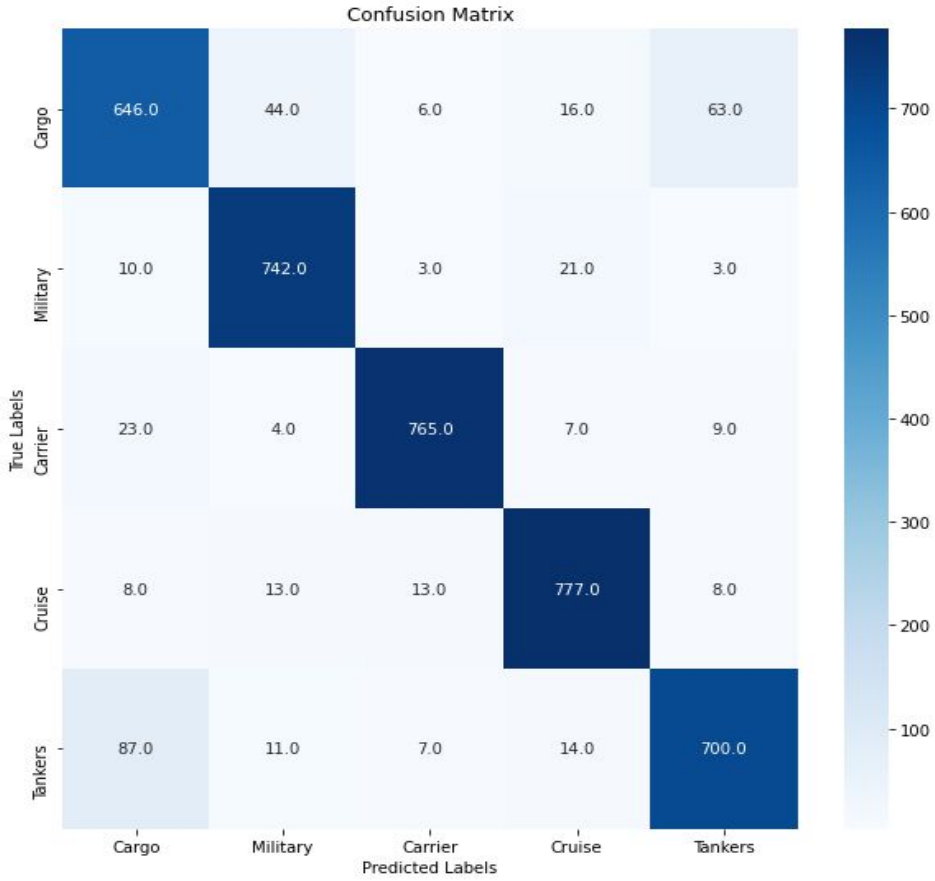
OTHER TRIALS

- Many trials have been made to further increase the success of the model.
- Additional layers was added to fully connected layers.
- Average Pooling was applied before passing from convolutional layers to fully connected layers.
- Then, batch normalization was applied to convolutional layers and fully connected layers after activation function.

- **Training Accuracy** : 99.92%
- **Validation Accuracy** : 92.97%

EVALUATION OF THE MODEL

- In the model we developed starting from the baseline, we reached 92.97% validation accuracy.
- We can see the prediction distribution of the classes in the confusion matrix of this model.



PRE-TRAINED MODELS

Models	Pool	E	LR	Training Accuracy	Validation Accuracy
ResNet50	max	50	0.000001	99.77%	86.47%
ResNet101	avg	25	0.000001	99.34%	92.57%
ResNet101	max	50	0.000001	99.89%	88.55%
ResNet50	avg	50	0.000001	99.88%	93.62%
VGG16	avg	50	0.000001	100%	93.19%
Xception	max	25	0.0001	99.80%	96.35%
Xception	avg	25	0.0001	99.88%	97.00%

CONCLUSIONS

- In this project, we built a classification model that classifies the ships in the dataset according to 5 different classes.
- Correctly tuning of the hyperparameters increase the success of the model.
- The deep networks learn more discriminative features as the layers go deeper.
- As our model gets deeper, we can increase the number of epocs.
- Creating a CNN model from scratch is a tiring and long process. Therefore, we can do improvements according to our dataset by using pre-trained models.

Thank you for listening.