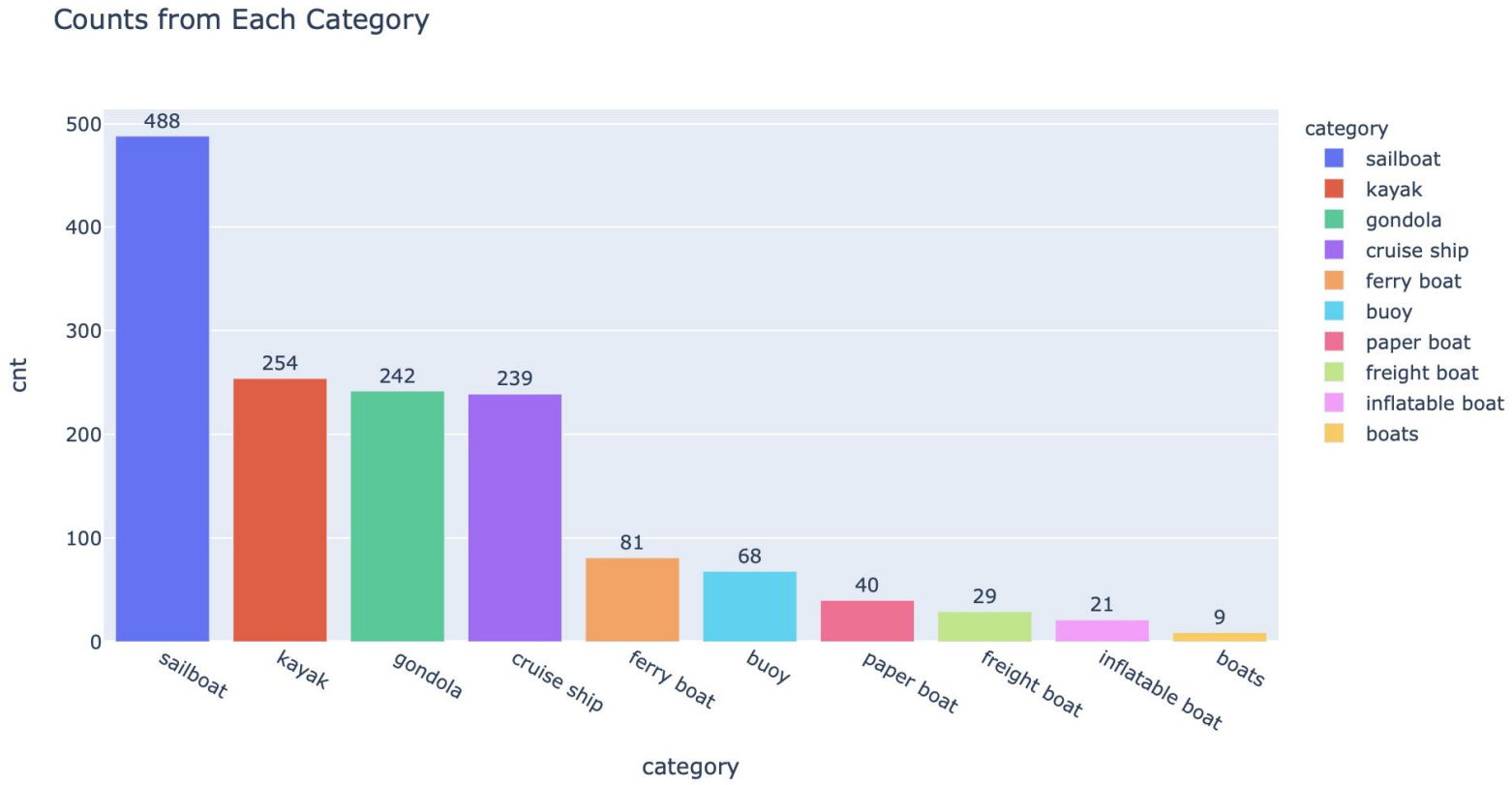
Multiclass Image Classification

Boat Types

Data Exploration:

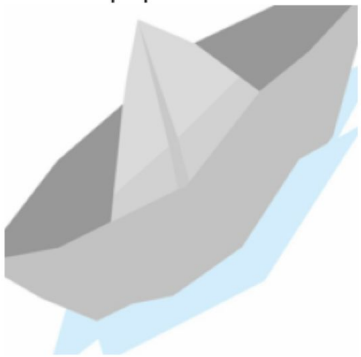


Sample of images from the dataset:

sailboat

cruise ship

paper boat



kayak

ferry boat

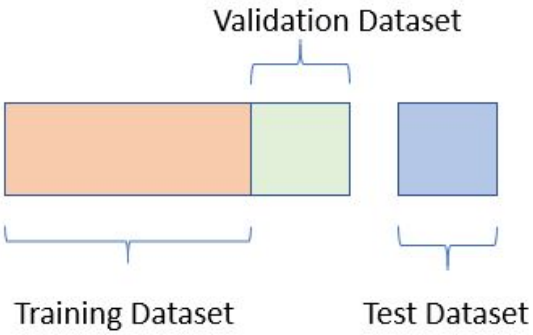
gondola

buoy

Pre-processing

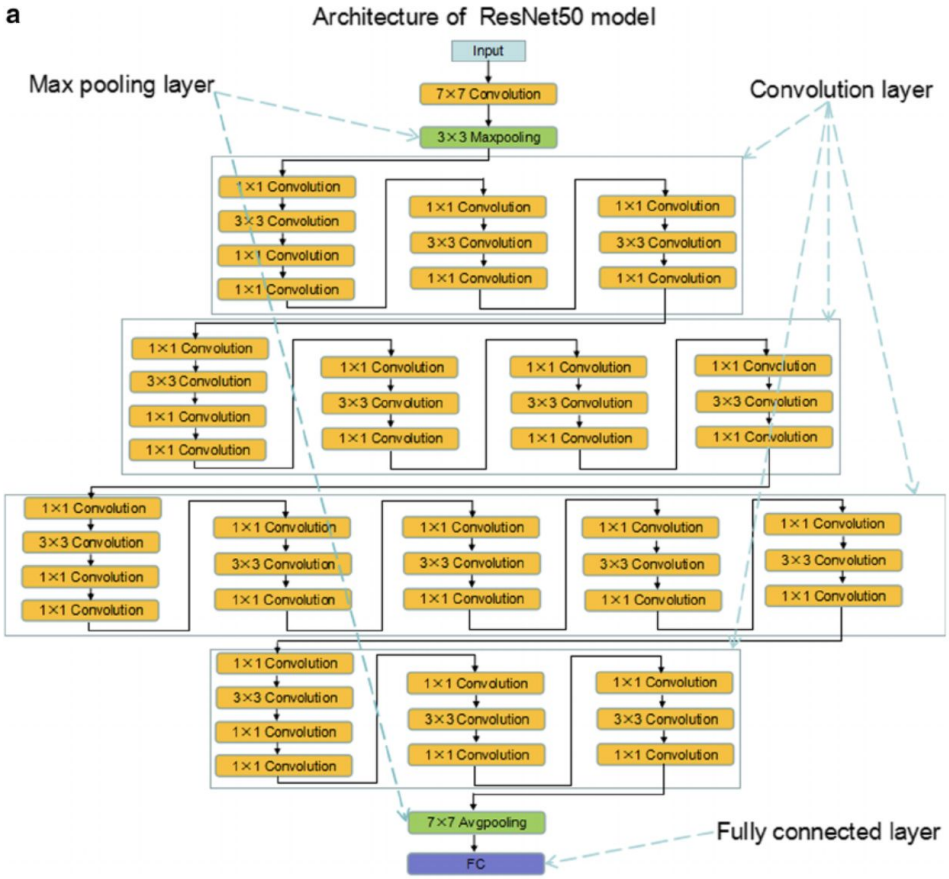
- **Image Transformations (ImageNet standards)**
    - Resize
        Resize input image to size = 256
    - Centercrop
        Crop the given image to size = 224
    - Transform to tensor
        Convert a PIL Image to tensor.
        features.shape ------ torch.Size([512, 3, 224, 224])
        labels.shape -------- torch.Size([512])
    - Normalize
        Normalize a tensor image with mean and standard deviation.

Create Train, Validation and Test Sets



	cat	train_cnt	val_cnt	test_cnt
0	buoy	41	13	14
1	cruise ship	143	48	48
2	ferry boat	49	16	16
3	gondola	145	49	48
4	kayak	152	51	51
5	paper boat	24	8	8
6	sailboat	293	98	97

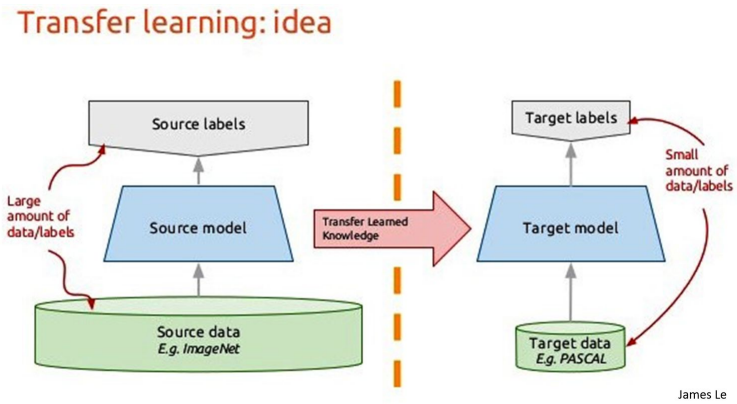
Model ResNet50



Description:

- CNN that is 50 layers deep.
- Trained on more than one million images from ImageNet database.
- Can classify images into 1000 object categories.
- Has learned rich feature representations from a wide range of images.
- Has an image input size of 224-by-224

Transfer Learning



- The main idea of TL is to implement a model quickly.
- Instead of creating a DNN from scratch, the model will transfer the features it has learned from the different dataset that has performed the same task.
- Also known as knowledge transfer.

Loss Function: NLLLOSS

The negative log likelihood loss. It is useful to train a classification problem with C classes.

Optimizer: ADAM

Implements Adam algorithm.

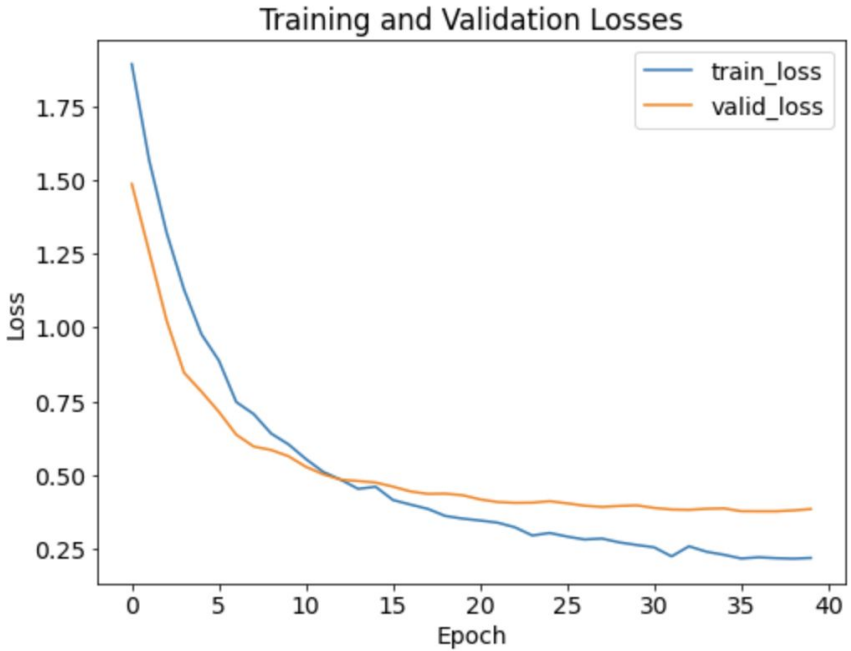
Replaced fully connected layer:

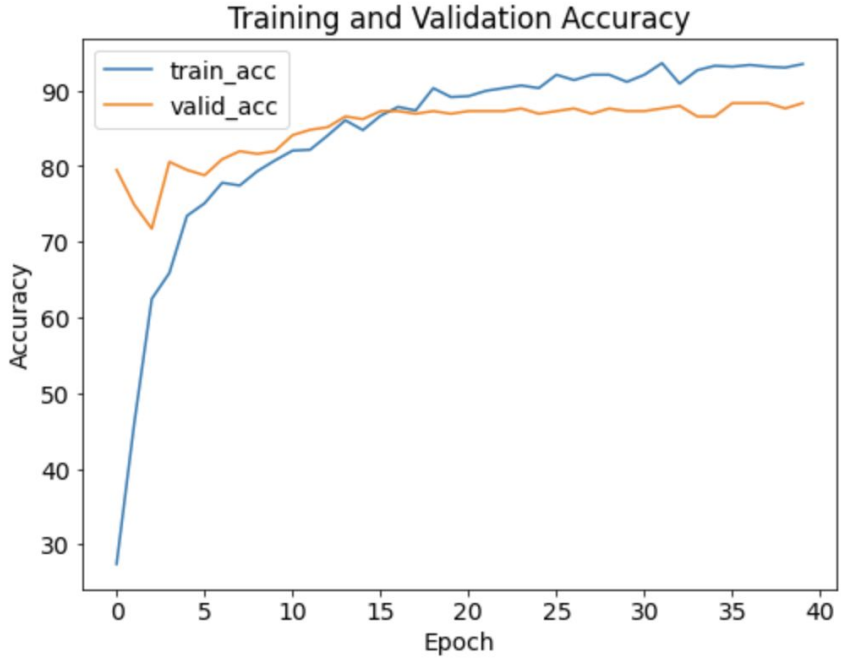
```
Sequential(
  (0): Linear(in_features=2048, out_features=256, bias=True)
  (1): ReLU()
  (2): Dropout(p=0.4, inplace=False)
  (3): Linear(in_features=256, out_features=7, bias=True)
  (4): LogSoftmax(dim=1)
)
```

Obtaining log-probabilities in a neural network is easily achieved by adding a LogSoftmax layer in the last layer of your network.

Loss & Accuracy:

Total epochs: 39. Best epoch: 36 with loss: 0.38 and accuracy: 88.34%

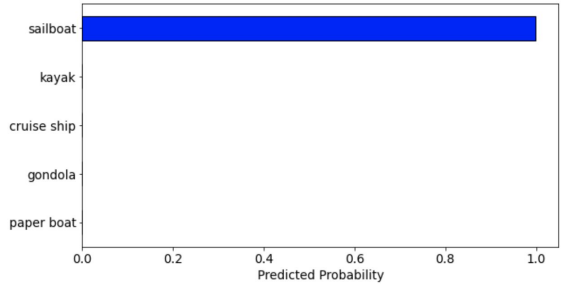


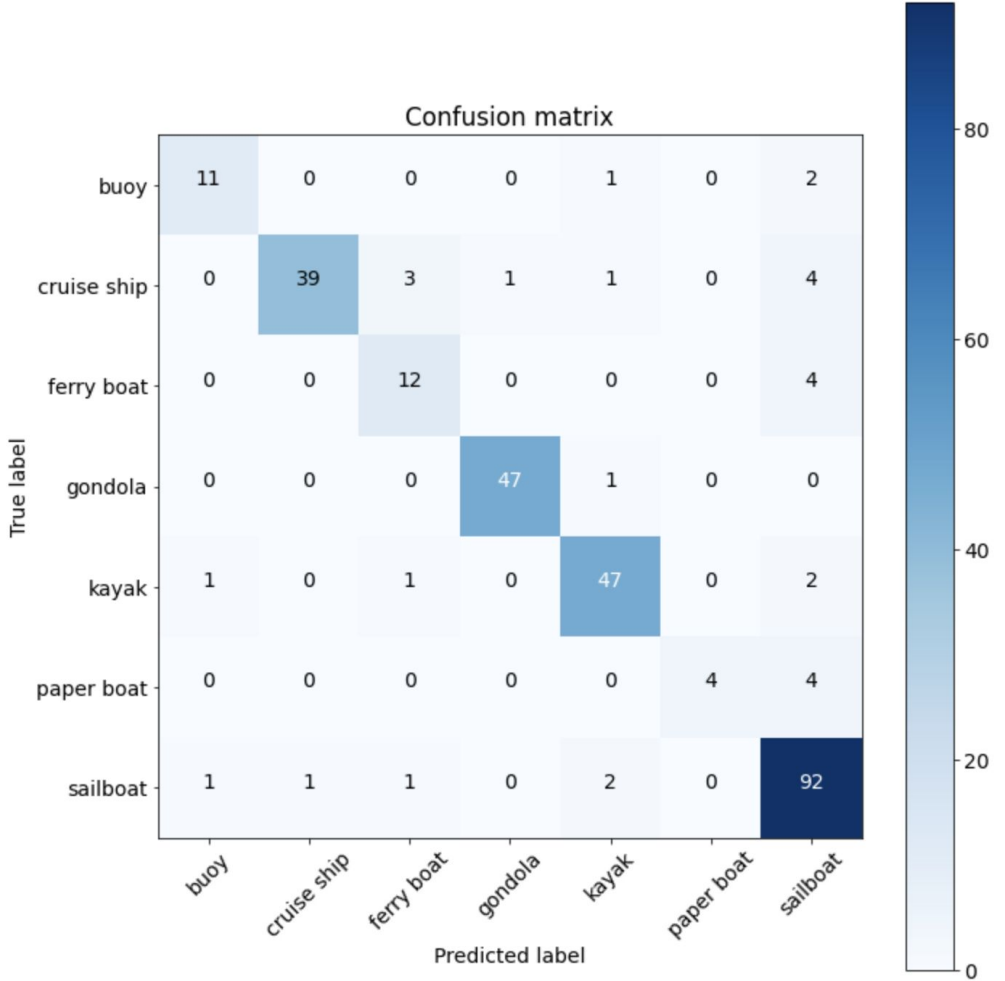


Test Results:

Overall Accuracy: 89%

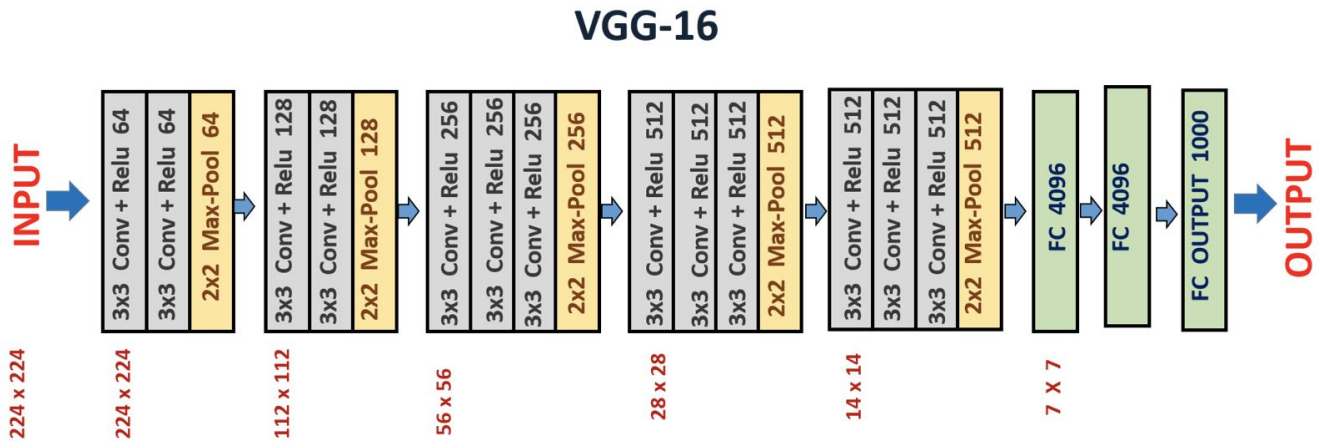
Prediction of a random test image





Model VGG16

Very deep convolutional networks for large-scale image recognition



Source : Google

- A stack of convolutional layers followed by three fully connected layers.
- All hidden layers are equipped with RELU.
- Max-pooling is performed over a 2-by-2 pixel window, with stride 2.

Loss Function: CROSS ENTROPY LOSS

Optimizer: SGD

Replaced fully connected layer:

```
Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=7, bias=True)
)
```

Loss & Accuracy:

Total epochs: 40. Best epoch: 38 with loss: 0.37 and accuracy: 89.63%

Test Results:

Overall Accuracy: 91%

Conclusion:

Overall the accuracy was similar on both models. The only difference was VGG16 took longer to train compared to ResNet.

In future try customizing the model by freezing and unfreezing layers, increasing the number of layers, and adjusting the learning rate.

References:

https://www.researchgate.net/figure/The-architecture-of-ResNet50-and-deep-learning-model-flowchart-a-b-Architecture-of_fig1_334767096
https://towardsdatascience.com/using-predefined-and-pretrained-cnns-in-pytorch-e3447cbe9e3c
https://www.pluralsight.com/guides/introduction-to-resnet
https://mlwhiz.medium.com
https://docs.openvinotoolkit.org/latest/omz_models_model_resnet_50_pytorch.html
https://arxiv.org/abs/1409.1556