Оптимизация RMSProp

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint
```

```
# creating the final model
model = Sequential()

model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = 'Same',
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same',
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))

filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]

image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
#Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop
epochs = 50
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = deca
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = 'categorical_crossentropy', optimizer = optimizer,
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
```

```
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:
```

```

Гипотеза о применении оптимизатора ADAM

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```

```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint


# creating the final model
model = Sequential()

model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))

filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]

image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
#Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop, Adam
epochs = 50
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
#optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = dec
optimizer = Adam(learning_rate = 0.001, epsilon = 1e-08, decay = decay)
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', met
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()
```

```
plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_2'], color='b', label="Training ROC")
plt.plot(history.history['val_auc_2'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_3'], color='b', label="Training PR")
plt.plot(history.history['val_auc_3'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()

plt.plot(history.history['auc_2'], color='b', label="Training ROC")
plt.plot(history.history['val_auc_2'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:
```
```

ADAM с иной функцией потерь

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
```

```
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint


# creating the final model
model = Sequential()

model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'Same', a
model.add(Conv2D(filters = 64, kernel_size = (3, 3), padding = 'Same', a
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 128, kernel_size = (3, 3), padding = 'Same',
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same',
model.add(Conv2D(filters = 256, kernel_size = (3, 3), padding = 'Same',
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same',
```

```
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', 
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', 
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', 
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', 
model.add(Conv2D(filters = 512, kernel_size = (3, 3), padding = 'Same', 
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))

filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]

image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
#Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop, Adam
epochs = 50
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
#optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = dec
optimizer = Adam(learning_rate = 0.001, epsilon = 1e-08, decay = decay)
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = "binary_crossentropy", optimizer = 'adam', metrics =
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
```

```
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:
```
```

RMSProp с иной функцией потерь и увеличением ядра свертки

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_size
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint
base_filter_count = 64
kernel = (5, 5)
# creating the final model
model = Sequential()

model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))

filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]

image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
    #Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop, Adam
epochs = 50
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = deca
#optimizer = Adam(learning_rate = 0.001, epsilon = 1e-08, decay = decay)
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = "binary_crossentropy", optimizer = optimizer, metri
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:

In [ ]:

RMSProp с иной функцией потерь и уменьшением ядра свертки

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint
base_filter_count = 64
kernel = (2, 2)
# creating the final model
model = Sequential()

model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))
```

```
filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]

image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
    #Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop, Adam
epochs = 50
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = deca
#optimizer = Adam(learning_rate = 0.001, epsilon = 1e-08, decay = decay)
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = "binary_crossentropy", optimizer = optimizer, metri
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
```

```
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:

RMSProp с иной функцией потерь и уменьшением ядра свертки, обучение 100 эпох

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
```

```
print(targets.shape)
del data_target
targets
```

In [ ]:
```
#Split Training data to training data and validate data to detect overfi
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint
base_filter_count = 64
kernel = (2, 2)
# creating the final model
model = Sequential()

model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))

filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]
```

```
image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
    #Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop, Adam
epochs = 100
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = deca
#optimizer = Adam(learning_rate = 0.001, epsilon = 1e-08, decay = decay)
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = "binary_crossentropy", optimizer = optimizer, metri
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()
```

```
plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:

RMSProp с иной функцией потерь и уменьшением ядра свертки, обучение 1000 эпох

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

```
#Split Training data to training data and validate data to detect overfi
```

In [ ]:
```
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(data,targets, test_siz
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

In [ ]:
```
#Data augumatation
from keras.preprocessing.image import ImageDataGenerator
img_gen = ImageDataGenerator()
```

In [ ]:
```
#Load ResNet50 model with Keras
#from keras.applications.vgg16 import VGG16 as PTModel, preprocess_input
#from keras.applications.densenet import DenseNet169 as PTModel, preproc
#from keras.applications.resnet50 import ResNet50 as ResModel
#from keras.applications.vgg16 import VGG16 as VGG16Model
#img_width, img_height = 256, 256
#model = VGG16Model(weights = 'imagenet', include_top=False, input_shape
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
#On this case, we only need predict 2 category (1. have ship, 2. no ship
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Bat
from keras.callbacks import ModelCheckpoint
base_filter_count = 64
kernel = (2, 2)
# creating the final model
model = Sequential()

model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(Conv2D(filters = base_filter_count, kernel_size = kernel, padd
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 2, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 4, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(Conv2D(filters = base_filter_count * 8, kernel_size = kernel,
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(1024, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(1024, activation = "relu"))
model.add(Dense(2, activation = "softmax"))

filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1
callbacks_list = [checkpoint]

image_file = 'model_1.png'
tf.keras.utils.plot_model(model, to_file = image_file, show_shapes = Tru
```

In [ ]:
```
    #Set Hyperparameter and Start training
from keras import optimizers
from keras.optimizers import RMSprop, Adam
```

```
epochs = 1000
lrate = 0.001 #learning rate
batch_size = 256
decay = lrate/epochs # Learning rate decay over each update
optimizer = RMSprop(lr = 0.001, rho = 0.9, epsilon = 1e-08, decay = deca
#optimizer = Adam(learning_rate = 0.001, epsilon = 1e-08, decay = decay)
#model.load_weights("transfer_ship_v1_5.h5")
model.compile(loss = "binary_crossentropy", optimizer = optimizer, metri
#model_final.summary()
```

In [ ]:
```
model.summary()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
history = model.fit(img_gen.flow(x_train, y_train, batch_size = batch_si
                                 steps_per_epoch = int(len(x_train)/batch_size)
model.save('transfer_ship_exp.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.show()
```

In [ ]:
```
gc.collect()
```

In [ ]:
```
plot_path = "/content/drive/MyDrive/Диплом/Ship_detection/CNN_PRACT/PLOT
plt.plot(history.history['loss'], color='b', label="Training loss")
plt.plot(history.history['val_loss'], color='r', label="validation loss"
plt.legend(loc='best', shadow=True)
plt.ylim(0, 50)
plt.savefig(plot_path + "loss.png")
plt.show()

plt.plot(history.history['accuracy'], color='b', label="Training accurac
plt.plot(history.history['val_accuracy'], color='r',label="Validation ac
legend = plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "acc.png")
plt.show()
```

```
plt.plot(history.history['auc'], color='b', label="Training ROC")
plt.plot(history.history['val_auc'], color='r',label="Validation ROC")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "roc.png")
plt.show()

plt.plot(history.history['auc_1'], color='b', label="Training PR")
plt.plot(history.history['val_auc_1'], color='r',label="Validation PR")
plt.legend(loc='best', shadow=True)
plt.savefig(plot_path + "pr.png")
plt.show()
```

In [ ]:

Базовое решение - подбрасывание монеты

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
import random
SEED = 42
```

In [ ]:
```
data = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input/data_
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Inpu
```

In [ ]:
```
print("array sizes of data array: ", data.shape)
print("array sizes of target array: ",data_target.shape)
print("example of one image in data array\n", data[0])
print("example of target for one image in array: ", data_target[0])
```

In [ ]:
```
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
n = len(targets)
n
```

In [ ]:
```
Y = np.array([[0] * 2] * n)
for i in range(n):
    rnd = random.random()
    Y[i] = [round(1 - rnd), round(rnd)]
Y
```

In [ ]:
```
from sklearn.metrics import accuracy_score
```

```
print("Точность (accuracy) равно ", accuracy_score(targets, Y))
```

In [ ]:
```
from sklearn.metrics import precision_score
print("Точность (precision) равно ",
      precision_score(targets.reshape(-1, 1), Y.reshape(-1, 1)))
```

In [ ]:
```
from sklearn.metrics import recall_score
print("Полнота (recall) равно ",
      recall_score(targets.reshape(-1, 1), Y.reshape(-1, 1)))
```

In [ ]:
```
from sklearn.metrics import precision_recall_curve
from sklearn import metrics
pr, rc, tr = precision_recall_curve(targets.reshape(-1, 1), np.array([1/
metrics.auc(pr, rc)
```

In [ ]:
```
```

Решение задачи человеком

In [ ]:
```
import tensorflow as tf
import os
import gc
import numpy as np
import pandas as pd
import time
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from PIL import Image
import random
SEED = 42
```

In [ ]:
```
data = pd.read_csv('/content/drive/MyDrive/Диплом/Ship_detection/Input/In
data_target = np.load('/content/drive/MyDrive/Диплом/Ship_detection/Input
#Set target to one hot target for classification problem
from sklearn.preprocessing import OneHotEncoder
targets = data_target.reshape(len(data_target),-1)
enc = OneHotEncoder()
enc.fit(targets)
targets = enc.transform(targets).toarray()
print(targets.shape)
del data_target
targets
```

In [ ]:
```
data
```

In [ ]:
```
Train_path = '/content/drive/MyDrive/Диплом/Ship_detection/Input/train/'
```

In [ ]:
```
%%time
index = 0
n = 100
Y = np.array([[0] * 2] * n)
Y_true = np.array([[0] * 2] * n)
l = data['ImageId'].values
anss = data['exist_ship'].values
offset = 500
for i in range(n):
    image_name = l[offset + i]
    imageA = Image.open(Train_path+image_name).resize((256, 256)) #open
```

```
    display(imageA)
    ans = int(input())
    Y[i][ans] = 1
    Y_true[i][anss[offset + i]] = 1
```

In [ ]:
```
np.save('/content/drive/MyDrive/Диплом/Ship_detection/Input/human_ans',
```

In [ ]:
```
from sklearn.metrics import accuracy_score
print("Точность (accuracy) равно ", accuracy_score(Y_true, Y))
```

In [ ]:
```
from sklearn.metrics import precision_score
print("Точность (precision) равно ",
      precision_score(Y_true.reshape(-1, 1), Y.reshape(-1, 1)))
```

In [ ]:
```
from sklearn.metrics import recall_score
print("Полнота (recall) равно ",
      recall_score(Y_true.reshape(-1, 1), Y.reshape(-1, 1)))
```