EE837 Project: Group 3

Jonas Nikula 20176392
Nikitas Ioannidis 20176445

1 Our idea

We wanted to use transfer learning and object detection in our project. Our original idea was to use a pretrained network, for example VGG16, and implement an object detection system on top off it. We wanted to do the object detection with YOLO [1], because it seemed like it would be simple enough to implement and train with our resources. Because of object detection, our dataset should be something that has multiple different object classes in it, PASCAL VOC for example.

However, we simply ran out of time. First we selected a much simpler dataset, the "Ships in Satellite Imagery" set from kaggle [2]. It only contains two classes, ship or no ship. It's size is only very limited. Our new plan was to train a classifier to detect the ships, and then add a much simpler version of YOLO on top of it. However, we only really completed the classifier, so in the end, our project is just on trying to classify the ships on the aforementioned dataset.

2 Implementation

We did the assignment using Python 3 and the Keras deep learning library. Training was done on the FloydHub cloud computing platform.

In addition to the codes and report, our zip folder contains the trained keras model, and the log file from training.

2.1 Team roles

We shared the work equally. The planning of the project was done together, and we consulted each other on trying to come up with the implementation.

2.2 Code details

The code itself is very simple. It's divided into 3 files.

`model.py`
`data.py`
`train_classifier.py`

The file `model.py` contains the actual model, and a function that returns the model. `train_classifier.py` instantiates and trains the model. `data.py` contains a helper function to read in the data set. The bash script `run_floydhub.sh` is just to make model training easier.

2.3 Model architecture and training

The model consists of 4 convolutional "blocks" and one dense block at the end. All convolutional layers use ReLU as the activation function. The last two blocks consist of a 3×3 layer, a 2×2 layer and a 1×1 layer. The first one is the same, but with 4 3×3 layers, and the second one has 2 3×3 layers.

At the end of a convolutional block there's a max pooling layer with a pool size of 2, and a dropout layer with a 25% chance of dropout. All the conv-layers use zero-padding, so that the output dimensions

are the same. The number of filters increases from 64 to 512, doubling at each block. The goal of this architecture was to get a lot of non-linearity, with a manageable parameter count.

The dense layer consists of 1024 unit layer, with ReLU activation and a 50% chance of dropout. The final layer has 1 unit with sigmoid activation. The Appendix A contains just the code to define the appendix, for an easy overview. The code should be clear enough to understand without knowledge of Python or Keras.

The dataset contains 2800 80×80 RGB images, which either contain or don't contain a ship. Training is done with a batch size of 100, for 20 epochs. The data loading function shuffles the data, from which we take 10% for validation. The data is also shuffled before each epoch. We use the adam optimizer, and the binary crossentropy loss function.

3 Performance

During training the model reported — at best — an accuracy of 0.9857 on the validation set. Training took about 10 minutes.

3.1 Analysis

During the creation of the model we tried to first use bigger filters at the beginning of the model, 7×7 on the first block, and 5×5 on the second. However, they did not improve performance and made training much slower. We opted to replace them with their equivalent filters; 4 and 2 3×3 layers respectively. Our final network is not only much more accurate, but also faster to train.

References

[1] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640.

[2] rhammell. (). Ships in satellite imagery, [Online]. Available: https://www.kaggle.com/rhammell/ships-in-satellite-imagery.

A Model code

```
## CONV_1 ########################################################################
    filter_count = 64
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(2,2), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(1,1), padding="same", activation="relu")(x)
    x = Dropout(0.25)(x)
    x = MaxPooling2D()(x)


## CONV_2 ########################################################################
    filter_count = 128
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(2,2), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(1,1), padding="same", activation="relu")(x)
    x = Dropout(0.25)(x)
    x = MaxPooling2D()(x)


## CONV_3 ########################################################################
    filter_count = 256
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(2,2), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(1,1), padding="same", activation="relu")(x)
    x = Dropout(0.25)(x)
    x = MaxPooling2D()(x)


## CONV_4 ########################################################################
    filter_count = 512
    x = Conv2D(filters=filter_count, kernel_size=(3,3), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(2,2), padding="same", activation="relu")(x)
    x = Conv2D(filters=filter_count, kernel_size=(1,1), padding="same", activation="relu")(x)
    x = Dropout(0.25)(x)
    x = MaxPooling2D()(x)


## DENSE_1 #######################################################################
    x = Flatten()(x)
    x = Dense(1024, activation="relu")(x)
    x = Dropout(0.5)(x)
    outputs = Dense(1, activation="sigmoid")(x)
```