Assignment 4: Dog-cat object detection

Aleksandra Jamróz* A.JAMROZ@STUDENTS.UU.NL
Karol Rogoziński* K.ROGOZINSKI@STUDENTS.UU.NL

1. Introduction

The overall goal of this task was to build and train an object detection network in the spirit of YOLOv1 for dog/cat head detection.

1.1. Data preparation

Dataset and dataloder templates were provided in the task description. The image size for this task was 112x112 and both images and bboxes were scaled using the following scales:

$$\text{scale}_x = \frac{\text{new}_w}{\text{orig}_w}, \quad \text{scale}_y = \frac{\text{new}_h}{\text{orig}_h} \tag{1}$$

The provided dataset (6) contained 3686 examples in total. The first 3000 were divided into train and valid using a stratified 80/20 split. The remaining 686 images were used as test data. Additionally, the training data set was augmented during training as described in 5.1.

Figure 1: Example images with bboxes from CatDogDataset

1.2. Base model

For the basic model, the architecture from the task description was followed. The exact architecture is attached in A. Architecture choices not described in assignment were motivated as follows:

- Activation Function: **ReLU** (Rectified Linear Unit) was chosen as the activation function because of its efficiency in deep networks. It helped mitigate the vanishing gradient problem and speeds up convergence.
- Pooling type: **Max Pooling** was used to reduce the spatial dimensions while preserving the most important features. It was chosen for the base model as the most common type of pooling.

Model weights are available in (1)

2. Training

Training was performed using the following hyperparameters:

- **Epochs**: 250
- **Learning Rate**: 1e-4
- **Batch Size**: 16

In addition, Adam was used as the optimiser and the learning rate was multiplied by 0.9 every 20 epochs to reach the optima.

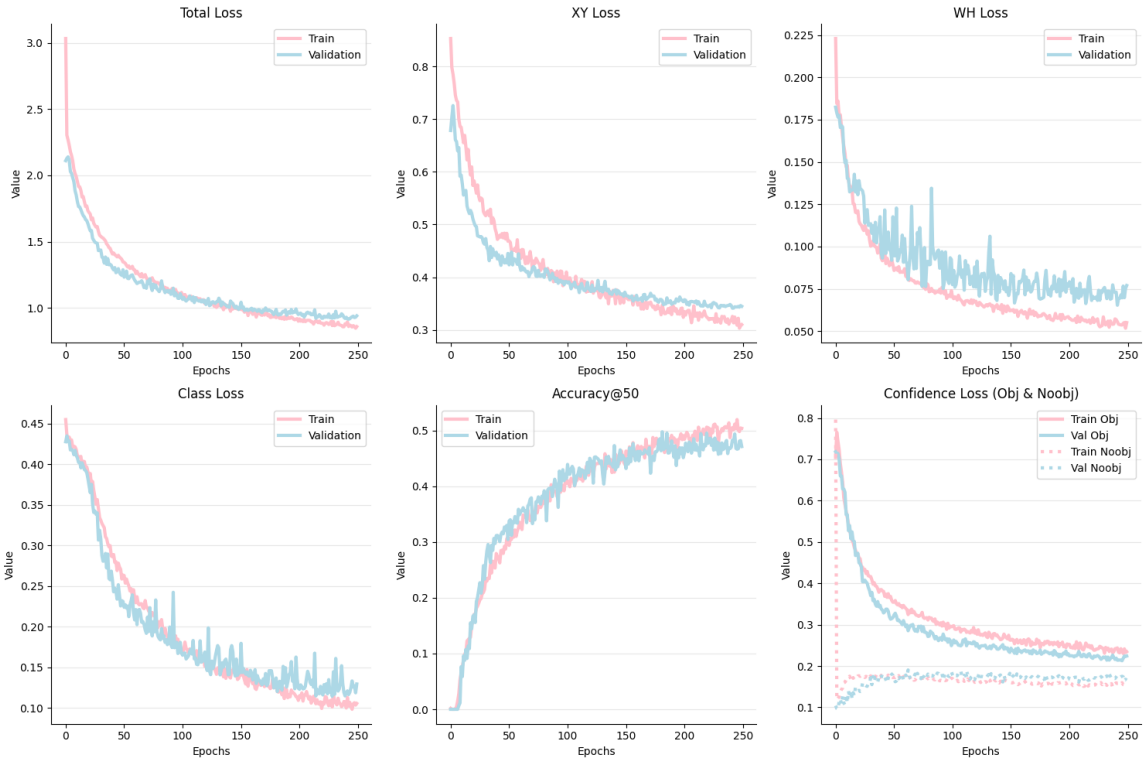

Figure 2: Visualisation of losses across training, both confidence losses are shown on the same graph for clarity.

3. Results

In addition to the base model, two others were trained: an improved version with modified architecture 5.2 and the one with ResNet18 (7) as the backbone 5.4. The results are shown in the table below 1:

Model	Train mAP	Validation mAP	Test mAP
CatDogYOLOv1	0.79	0.74	0.65
CatDogYOLOv1Improved	0.89	**0.83**	0.77
CatDogYOLOv1ResBB	**0.92**	0.80	**0.81**

Table 1: The best Mean Average Precision (mAP) for different models.

All the results presented are obtained with the extended data. Without it, they were all significantly lower. The confusion matrices are shown below. 3

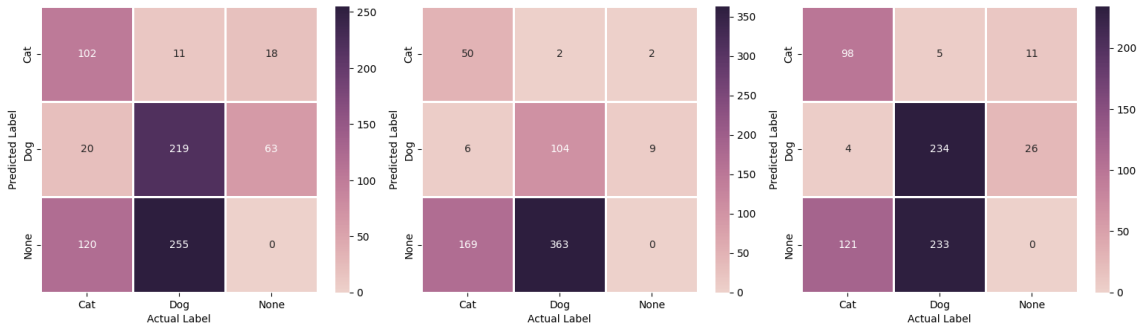

Figure 3: Confusion matrix for the best objective threshold for each model, from left: base, improved, resnetbb

Below 4 are presented some predictions of the best model:

Figure 4: Example predictions of the model with resnet backbone

4. Conclusions

The results for all models are quite solid. The thing worth noting is that the models are often not sure about the prediction and therefore a lot of entities are not classified at all. This could probably be improved with longer training, as the training plots show that the models probably do not have full coverage yet. There is also an imbalance between cats and dogs, especially in the video. This could probably be improved by balancing the training data sets.

5. Additional Improvements, aka Choice Tasks

5.1. Data augmentation

The same techniques were used to augment the data as in the original paper:

- **Random Scaling** $(0.8 : 1.2)$, applies a transformation matrix to scale the image and bounding boxes:

$$T = \begin{bmatrix} s & 0 & (1-s) \\ \cdot c_x & 0 & s & (1-s) \cdot c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

  where s is a random scale factor, and c is the image center.

- **Random Translation** $(-0.2 : 0.2)$ to shift the image and bounding boxes:

$$t_x = r_x \cdot w, \quad t_y = r_y \cdot h \tag{3}$$

  where r are random values in a predefined range.

- **Random Brightness and Saturation** (3) applied by multiplying pixel values by a factor:

$$f = \text{random}\left(\frac{1}{e}, e\right) \tag{4}$$

  where is the exposure factor. This modifies the image pixel intensities without affecting bboxes.

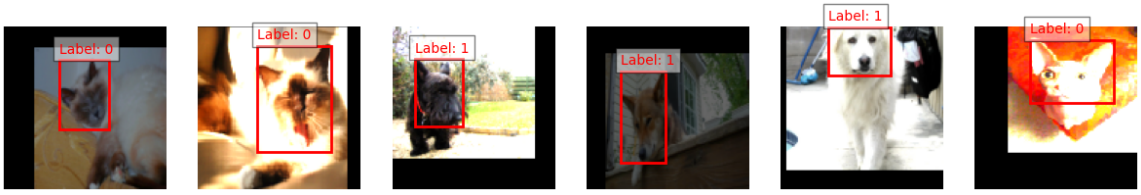

Figure 5: Example of augmented images

5.2. Improved YOLOv1

In addition to the basic model, a model with some improvements was proposed. Changes included:

- The activation function was changed from ReLU to LeakyRELU with a negative range of 0.1, as in the original paper, to avoid vanishing gradients.
- An additional dropout was added between the last two fully connected layers to prevent overfitting along with the data augmentation
- An additional convolution layer was added instead of the last pooling to extract more spatial information from the image

Model weights are available in (2) and training losses are shown below 6.

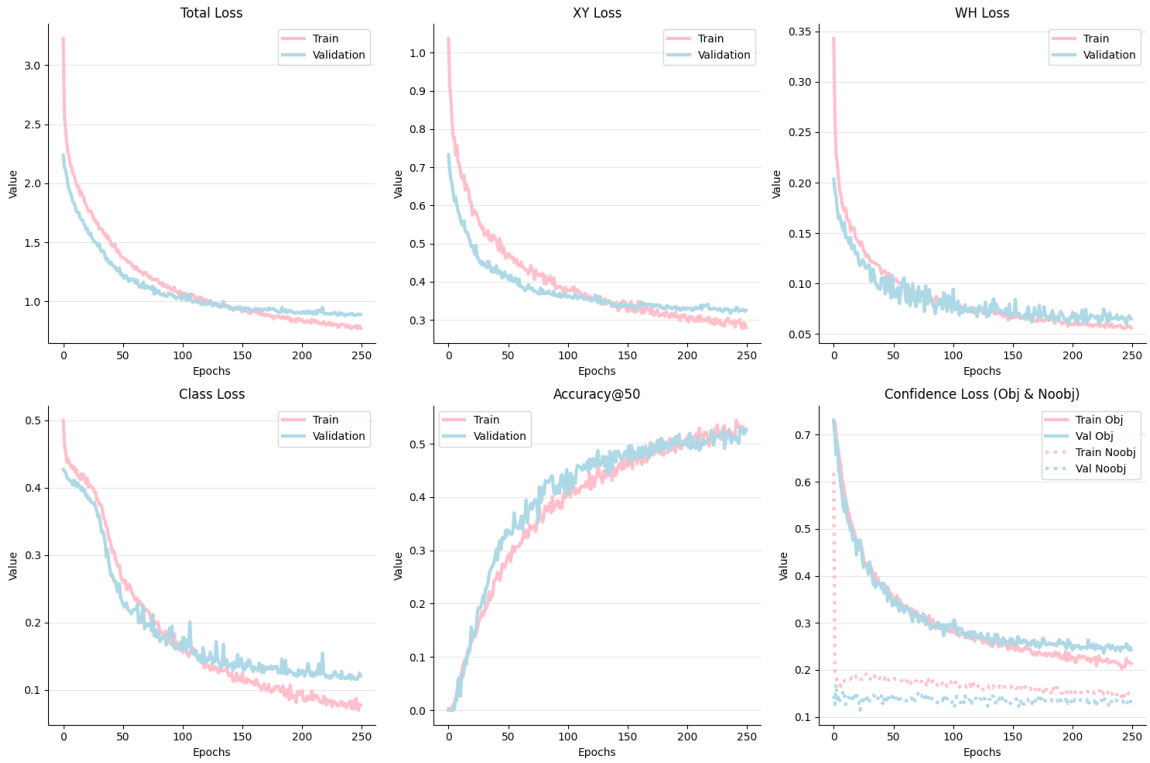

Figure 6: Visualisation of losses over training for the improved model.

5.3. Prediction on the video

The prediction on a video was performed as an additional test case. The result can be seen in (4).

5.4. YOLOv1 with ResNet18 Backbone

For the final model, the pre-trained ResNet 18 (7) was used as the backbone instead of the convolutional layers. However, the initial results were very poor, so the last 10 layers were left unfrozen to improve performance.

Model weights are available in (3) and training losses are shown below 7.

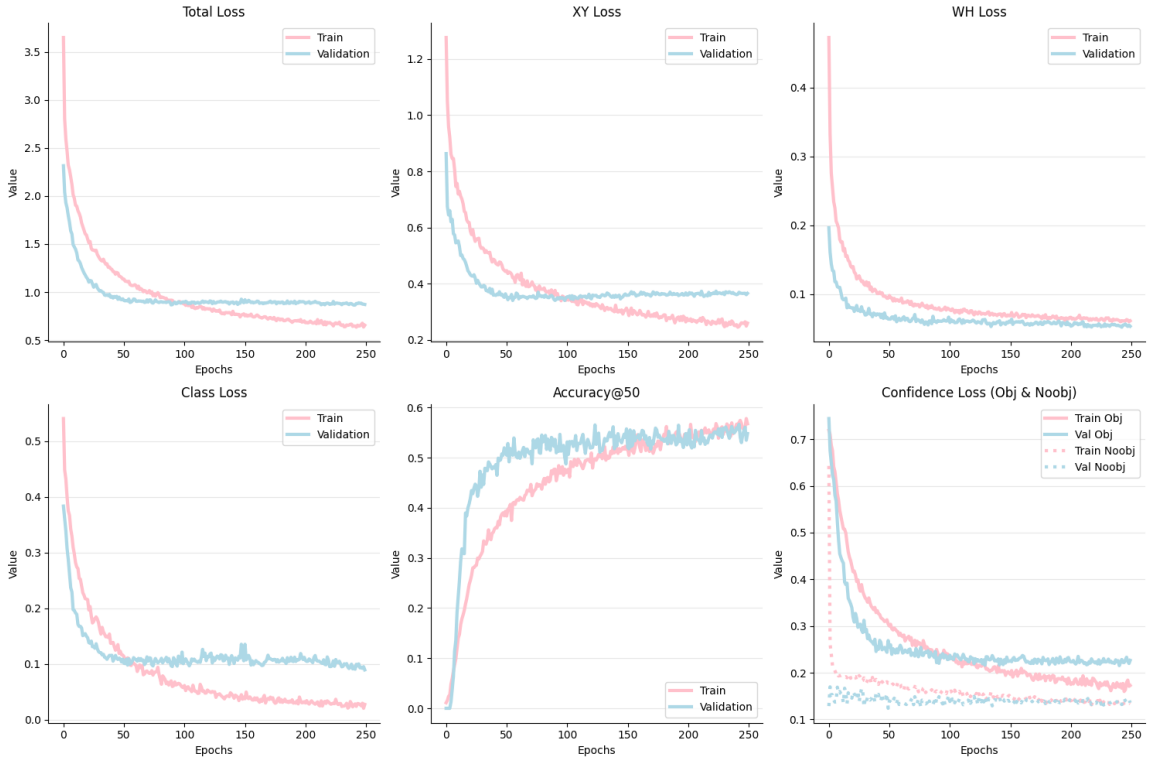

Figure 7: Visualisation of losses over training for the resnet model.

References

[1] Simple YOLOv1 Weights. Available: Google Drive

[2] Improved YOLOv1 Weights. Available: Google Drive

[3] YOLOv1 with ResNet18 Backbone Weights. Available: Google Drive

[4] Video with predictions. Available: Google Drive

[5] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[6] Kaggle. (n.d.). Dog and Cat Detection." Retrieved from https://www.kaggle.com/datasets/andrewmvd/dog-and-cat-detection.

[7] He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep Residual Learning for Image Recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Appendix A. Basic YOLOv1 Architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 112, 112]             448
       BatchNorm2d-2         [-1, 16, 112, 112]              32
         MaxPool2d-3           [-1, 16, 56, 56]               0
            Conv2d-4           [-1, 32, 56, 56]           4,640
       BatchNorm2d-5           [-1, 32, 56, 56]              64
         MaxPool2d-6           [-1, 32, 28, 28]               0
            Conv2d-7           [-1, 64, 28, 28]          18,496
       BatchNorm2d-8           [-1, 64, 28, 28]             128
         MaxPool2d-9           [-1, 64, 14, 14]               0
           Conv2d-10           [-1, 64, 14, 14]          36,928
      BatchNorm2d-11           [-1, 64, 14, 14]             128
        MaxPool2d-12             [-1, 64, 7, 7]               0
           Conv2d-13             [-1, 32, 7, 7]          18,464
      BatchNorm2d-14             [-1, 32, 7, 7]              64
          Flatten-15                 [-1, 1568]               0
          Dropout-16                 [-1, 1568]               0
           Linear-17                  [-1, 512]         803,328
           Linear-18                  [-1, 343]         175,959
================================================================
Total params: 1,058,679
Trainable params: 1,058,679
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.14
Forward/backward pass size (MB): 6.30
Params size (MB): 4.04
Estimated Total Size (MB): 10.48
----------------------------------------------------------------
```

Appendix B. Improved YOLOv1 Architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 112, 112]             448
       BatchNorm2d-2         [-1, 16, 112, 112]              32
         MaxPool2d-3           [-1, 16, 56, 56]               0
            Conv2d-4           [-1, 32, 56, 56]           4,640
       BatchNorm2d-5           [-1, 32, 56, 56]              64
         MaxPool2d-6           [-1, 32, 28, 28]               0
            Conv2d-7           [-1, 64, 28, 28]          18,496
       BatchNorm2d-8           [-1, 64, 28, 28]             128
         MaxPool2d-9           [-1, 64, 14, 14]               0
           Conv2d-10           [-1, 64, 14, 14]          36,928
      BatchNorm2d-11           [-1, 64, 14, 14]             128
        MaxPool2d-12             [-1, 64, 7, 7]               0
           Conv2d-13             [-1, 64, 7, 7]          36,928
      BatchNorm2d-14             [-1, 64, 7, 7]             128
           Conv2d-15            [-1, 128, 7, 7]          73,856
      BatchNorm2d-16            [-1, 128, 7, 7]             256
          Flatten-17                 [-1, 6272]               0
          Dropout-18                 [-1, 6272]               0
           Linear-19                  [-1, 512]       3,211,776
          Dropout-20                  [-1, 512]               0
           Linear-21                  [-1, 343]         175,959
================================================================
Total params: 3,559,767
Trainable params: 3,559,767
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.14
Forward/backward pass size (MB): 6.49
Params size (MB): 13.58
Estimated Total Size (MB): 20.22
----------------------------------------------------------------
```

Appendix C. YOLOv1 with ResNet18 Backbone Architecture

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 56, 56]	9,408
BatchNorm2d-2	[-1, 64, 56, 56]	128
ReLU-3	[-1, 64, 56, 56]	0
MaxPool2d-4	[-1, 64, 28, 28]	0
Conv2d-5	[-1, 64, 28, 28]	36,864
BatchNorm2d-6	[-1, 64, 28, 28]	128
ReLU-7	[-1, 64, 28, 28]	0
Conv2d-8	[-1, 64, 28, 28]	36,864
BatchNorm2d-9	[-1, 64, 28, 28]	128
ReLU-10	[-1, 64, 28, 28]	0
BasicBlock-11	[-1, 64, 28, 28]	0
Conv2d-12	[-1, 64, 28, 28]	36,864
BatchNorm2d-13	[-1, 64, 28, 28]	128
ReLU-14	[-1, 64, 28, 28]	0
Conv2d-15	[-1, 64, 28, 28]	36,864
BatchNorm2d-16	[-1, 64, 28, 28]	128
ReLU-17	[-1, 64, 28, 28]	0
BasicBlock-18	[-1, 64, 28, 28]	0
Conv2d-19	[-1, 128, 14, 14]	73,728
BatchNorm2d-20	[-1, 128, 14, 14]	256
ReLU-21	[-1, 128, 14, 14]	0
Conv2d-22	[-1, 128, 14, 14]	147,456
BatchNorm2d-23	[-1, 128, 14, 14]	256
Conv2d-24	[-1, 128, 14, 14]	8,192
BatchNorm2d-25	[-1, 128, 14, 14]	256
ReLU-26	[-1, 128, 14, 14]	0
BasicBlock-27	[-1, 128, 14, 14]	0
Conv2d-28	[-1, 128, 14, 14]	147,456
BatchNorm2d-29	[-1, 128, 14, 14]	256
ReLU-30	[-1, 128, 14, 14]	0
Conv2d-31	[-1, 128, 14, 14]	147,456
BatchNorm2d-32	[-1, 128, 14, 14]	256
ReLU-33	[-1, 128, 14, 14]	0
BasicBlock-34	[-1, 128, 14, 14]	0
Conv2d-35	[-1, 256, 7, 7]	294,912
BatchNorm2d-36	[-1, 256, 7, 7]	512
ReLU-37	[-1, 256, 7, 7]	0
Conv2d-38	[-1, 256, 7, 7]	589,824
BatchNorm2d-39	[-1, 256, 7, 7]	512
Conv2d-40	[-1, 256, 7, 7]	32,768

```
      BatchNorm2d-41            [-1, 256, 7, 7]             512
             ReLU-42            [-1, 256, 7, 7]               0
       BasicBlock-43            [-1, 256, 7, 7]               0
           Conv2d-44            [-1, 256, 7, 7]         589,824
      BatchNorm2d-45            [-1, 256, 7, 7]             512
             ReLU-46            [-1, 256, 7, 7]               0
           Conv2d-47            [-1, 256, 7, 7]         589,824
      BatchNorm2d-48            [-1, 256, 7, 7]             512
             ReLU-49            [-1, 256, 7, 7]               0
       BasicBlock-50            [-1, 256, 7, 7]               0
           Conv2d-51            [-1, 512, 4, 4]       1,179,648
      BatchNorm2d-52            [-1, 512, 4, 4]           1,024
             ReLU-53            [-1, 512, 4, 4]               0
           Conv2d-54            [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-55            [-1, 512, 4, 4]           1,024
           Conv2d-56            [-1, 512, 4, 4]         131,072
      BatchNorm2d-57            [-1, 512, 4, 4]           1,024
             ReLU-58            [-1, 512, 4, 4]               0
       BasicBlock-59            [-1, 512, 4, 4]               0
           Conv2d-60            [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-61            [-1, 512, 4, 4]           1,024
             ReLU-62            [-1, 512, 4, 4]               0
           Conv2d-63            [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-64            [-1, 512, 4, 4]           1,024
             ReLU-65            [-1, 512, 4, 4]               0
       BasicBlock-66            [-1, 512, 4, 4]               0
          Dropout-67                [-1, 8192]               0
           Linear-68                 [-1, 512]       4,194,816
          Dropout-69                 [-1, 512]               0
           Linear-70                 [-1, 343]         175,959
================================================================
Total params: 15,547,287
Trainable params: 6,732,119
Non-trainable params: 8,815,168
----------------------------------------------------------------
Input size (MB): 0.14
Forward/backward pass size (MB): 16.00
Params size (MB): 59.31
Estimated Total Size (MB): 75.45
----------------------------------------------------------------
```