

Module 4: Using an “Off-the-shelf” Model

Trang Tran

CPS, Northeastern University

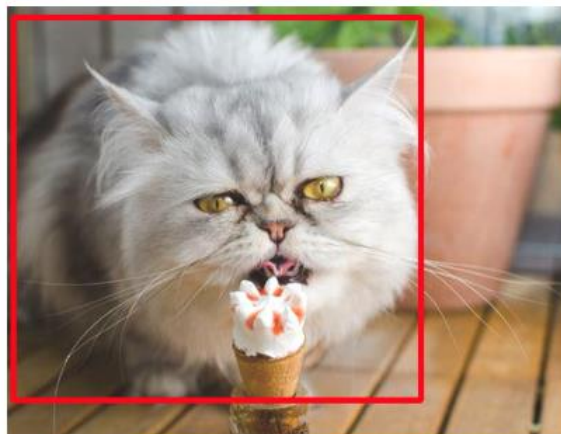
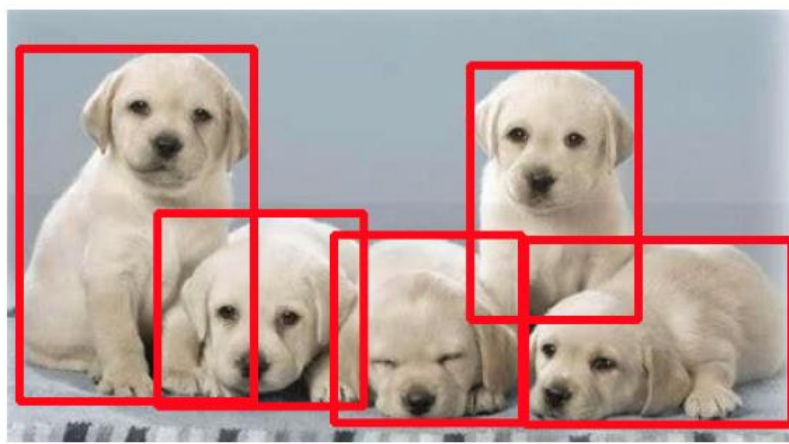
EAI 6010 | Applications of Artificial Intelligence

Dr. Dustin Garvey

June 18, 2024

Data Selection and Applications

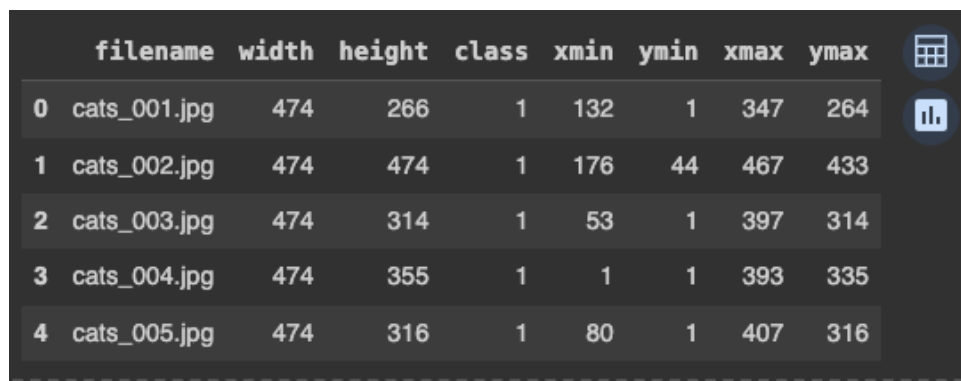
The dataset was selected based on its relevance and suitability for the task of object detection involving common animals such as dogs, cats, and monkeys. I chose the Torch Vision Object Detection fine-tuning tutorial^[1] and sourced this small dataset from Kaggle^[2]. It contains images of three different animal species, providing a good variety for training an object detection model. The dataset includes hand-annotated images with annotations available in XML and CSV formats. With 469 images in the training set and 51 images in the test set, it is large enough to fine-tune a model but small enough to manage with available computational resources. The dataset can be used as a starting point for learning about training object detection using transfer learning.



Adaptations for Use

To use the dataset effectively, several modifications and data preparations were necessary.

- Handling the annotation files and encoding the classes to integers
- Data augmentation: To enhance the model's generalization ability, data augmentation techniques such as random cropping, flipping, and color adjustments were applied to the training images.
- Format consistency and setting up Torch dataset: Ensuring consistency in image path and formats, folder structure, and resizing images to a standard dimension suitable for the model architecture.



	filename	width	height	class	xmin	ymin	xmax	ymax
0	cats_001.jpg	474	266	1	132	1	347	264
1	cats_002.jpg	474	474	1	176	44	467	433
2	cats_003.jpg	474	314	1	53	1	397	314
3	cats_004.jpg	474	355	1	1	1	393	335
4	cats_005.jpg	474	316	1	80	1	407	316

Challenges and Solutions

One of my primary difficulties was selecting a dataset that meets all the requirements of the object detection problem, according to the Pytorch tutorial. Subsequently, I stumbled upon data transformation and the creation of a Torch dataset from the original one. To be honest, it was overwhelming for me to imagine the parameters of this learning problem, and the process was harsh. I've tried several datasets and failed to load and train them. A preprocessing pipeline was implemented to reformat all images and annotations, ensuring uniformity. The process ensures that each image and its corresponding annotations are prepared in a format compatible with the requirements of object detection models, enabling seamless integration into the training pipeline.

Another challenge was the limited number of images in the dataset, which could lead to overfitting. To mitigate this, data augmentation and transforms were employed.

Model Deployment Considerations

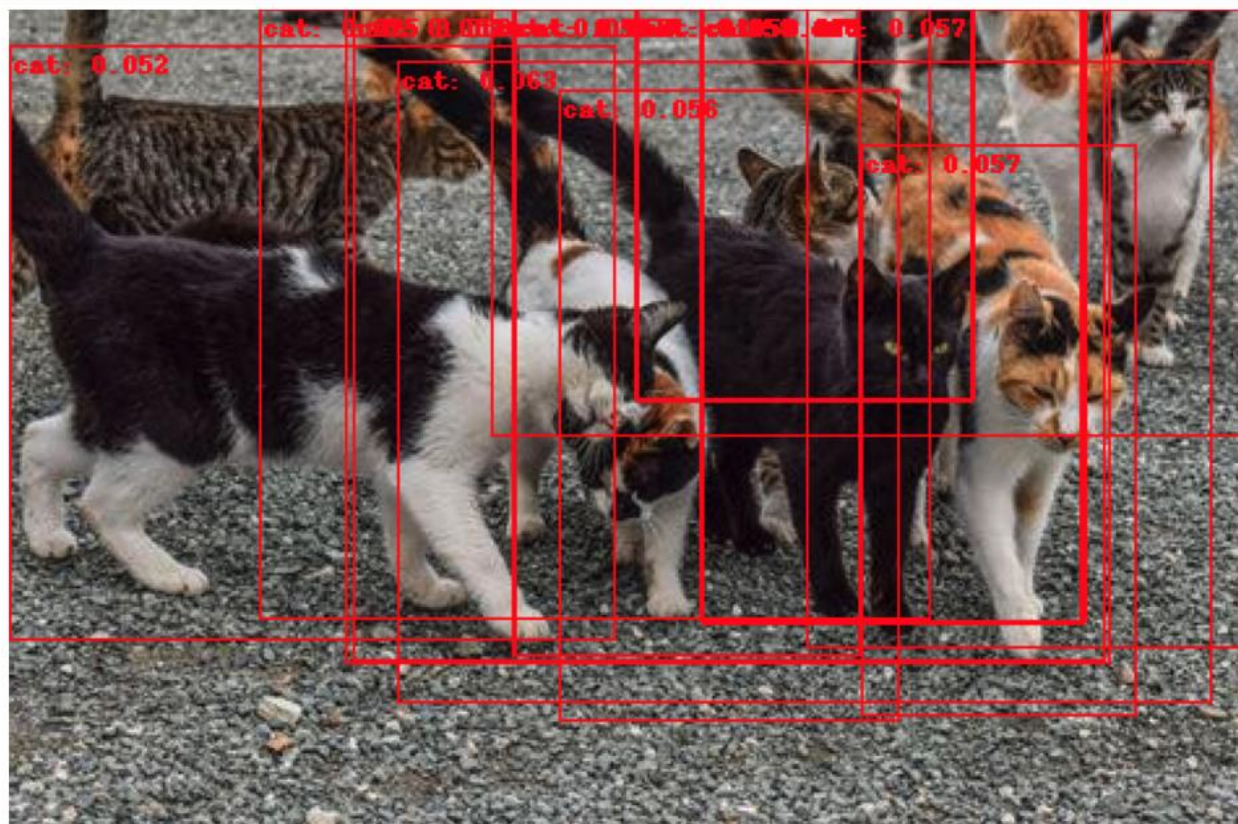
The tutorial implementation initially produced a functional model but required further refinement for deployment readiness. With training for multiple epochs, I encountered persistent issues with the 'evaluation' process, preventing me from obtaining evaluation scores on the test set.

Additionally, I require more guidance on effectively assessing the model's performance in terms of object detection accuracy, loss metrics, and its ability to generalize.

Further improvement strategies need to be applied for hyperparameter tuning, and cross-validation to enhance model performance. It's clear that achieving satisfactory outcomes will require iterative fine-tuning and continuous improvement efforts.

```
Epoch: [0] [ 0/118] eta: 0:10:14 lr: 0.000048 loss: 2.3377 (2.3377) loss_classifier: 1.3242 (1.3242) loss_box_reg: 0.2144 (0.2144) loss_objectness: 0.7012 (0.7012) loss_rpn_box_reg: 0.0979 (0.0979)
Epoch: [0] [10/118] eta: 0:02:18 lr: 0.000475 loss: 2.1833 (2.2360) loss_classifier: 1.2757 (1.2407) loss_box_reg: 0.0935 (0.0982) loss_objectness: 0.6951 (0.6987) loss_rpn_box_reg: 0.1587 (0.1983)
Epoch: [0] [20/118] eta: 0:01:41 lr: 0.000902 loss: 2.0429 (1.8958) loss_classifier: 1.0145 (0.9431) loss_box_reg: 0.1002 (0.1094) loss_objectness: 0.6891 (0.6776) loss_rpn_box_reg: 0.1480 (0.1657)
Epoch: [0] [30/118] eta: 0:01:21 lr: 0.001328 loss: 1.2908 (1.6857) loss_classifier: 0.3916 (0.7566) loss_box_reg: 0.1133 (0.1253) loss_objectness: 0.6184 (0.6390) loss_rpn_box_reg: 0.1170 (0.1648)
Epoch: [0] [40/118] eta: 0:01:07 lr: 0.001755 loss: 1.2239 (1.6070) loss_classifier: 0.3973 (0.6988) loss_box_reg: 0.1808 (0.1502) loss_objectness: 0.5142 (0.6000) loss_rpn_box_reg: 0.1479 (0.1579)
Epoch: [0] [50/118] eta: 0:00:55 lr: 0.002182 loss: 1.2100 (1.5200) loss_classifier: 0.4065 (0.6373) loss_box_reg: 0.2050 (0.1658) loss_objectness: 0.4400 (0.5632) loss_rpn_box_reg: 0.1443 (0.1537)
Epoch: [0] [60/118] eta: 0:00:46 lr: 0.002609 loss: 1.1089 (1.4567) loss_classifier: 0.3669 (0.5972) loss_box_reg: 0.2050 (0.1737) loss_objectness: 0.3868 (0.5331) loss_rpn_box_reg: 0.1405 (0.1527)
Epoch: [0] [70/118] eta: 0:00:37 lr: 0.003036 loss: 1.1089 (1.4047) loss_classifier: 0.3314 (0.5630) loss_box_reg: 0.2233 (0.1855) loss_objectness: 0.3542 (0.5046) loss_rpn_box_reg: 0.1405 (0.1517)
Epoch: [0] [80/118] eta: 0:00:28 lr: 0.003463 loss: 1.0791 (1.3624) loss_classifier: 0.3314 (0.5398) loss_box_reg: 0.2233 (0.1889) loss_objectness: 0.3182 (0.4799) loss_rpn_box_reg: 0.1493 (0.1538)
Epoch: [0] [90/118] eta: 0:00:20 lr: 0.003890 loss: 0.9665 (1.3184) loss_classifier: 0.3255 (0.5151) loss_box_reg: 0.2162 (0.1928) loss_objectness: 0.3038 (0.4609) loss_rpn_box_reg: 0.1223 (0.1496)
Epoch: [0] [100/118] eta: 0:00:13 lr: 0.004317 loss: 0.9557 (1.2888) loss_classifier: 0.3315 (0.4998) loss_box_reg: 0.2535 (0.2030) loss_objectness: 0.2520 (0.4396) loss_rpn_box_reg: 0.1162 (0.1463)
Epoch: [0] [110/118] eta: 0:00:05 lr: 0.004744 loss: 0.9778 (1.2654) loss_classifier: 0.3518 (0.4873) loss_box_reg: 0.2961 (0.2126) loss_objectness: 0.2391 (0.4223) loss_rpn_box_reg: 0.1160 (0.1432)
Epoch: [0] [117/118] eta: 0:00:00 lr: 0.005000 loss: 0.9932 (1.2442) loss_classifier: 0.3373 (0.4765) loss_box_reg: 0.2935 (0.2142) loss_objectness: 0.2277 (0.4103) loss_rpn_box_reg: 0.1160 (0.1432)
Epoch: [0] Total time: 0:01:23 (0.7073 s / it)
creating index...
index created!
```

During testing, the model exhibited poor performance, especially when applied to images containing groups of animals (see picture below). As a result, I do not feel confident deploying this model without addressing these challenges. Further exploration and potential alternative solutions are necessary to overcome these complexities.



References

1. TorchVision Object Detection Finetuning Tutorial.
https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
2. <https://www.kaggle.com/code/tarunbisht11/torchvision-object-detection>