YOLO: You Only Look Once

SERPRO

Quem sou?

- Analista no SERPRO desde 2010 atualmente lotado na DINEF/SUPAN/ANCEI/ANSVB

- Mestre pela UFRGS em 2022: "Estudo de impacto de técnicas de IA e comunicação para aplicações edge"

Algoritmo You Only Look Once (Yolo)

- O YOLO foi introduzido por **Joseph Redmon**, juntamente com Santosh Divvala, Ross Girshick e Ali Farhadi, em um artigo publicado em **2015**: *"You Only Look Once: Unified, Real-Time Object Detection"*.

- O YOLO (You Only Look Once) é um algoritmo de detecção de objetos em tempo real que divide uma imagem em uma grade e, em cada célula, prevê caixas delimitadoras (bounding boxes) e suas respectivas classes.

Algoritmo You Only Look Once (Yolo)

- Ao contrário de métodos tradicionais que analisam a imagem em múltiplas etapas ou regiões, o YOLO realiza todas as previsões em uma única passagem pela rede, tornando-o extremamente rápido

- Darknet V3 ("Jazz") lançado em Outubro de 2024 pode processar o LEGO dataset videos a **1000 FPS** usando a NVIDIA RTX 3090 GPU, ou seja os frames são recebidos, analizados e processados pela rede Darknet/YOLO em 1 milisegundo ou menos

Algoritmo You Only Look Once (Yolo)

- Versões:
- YOLOv1 (2016): primeiro modelo YOLO, foco na velocidade
- YOLOv2 (2017): melhoria na acurácia e introdução de classes de objetos
- YOLOv3 (2018): permitiu detecção em escalas variadas
- YOLOv4 (2020): melhorias no modelo saída de Redmon
- YOLOv7 (2022): melhoria da precisão, aplicações de tempo real
- YOLO-NAS(2023): focado em redes neurais adaptativas

Considerações sobre performance

YOLO Version	GPU Model	FPS (approx.)	Resolution
YOLOv5 (small)	NVIDIA RTX 3080	~140 FPS	640x640
YOLOv5 (x-large)	NVIDIA RTX 3090	~100 FPS	1280x1280
YOLOv6	NVIDIA A100	~175 FPS	640x640

Considerações sobre performance

YOLOv7	NVIDIA V100	30-160 FPS	Variable
YOLOv8 (nano)	NVIDIA RTX 4070 Ti	~200 FPS	640x640
YOLOv8 (large)	NVIDIA RTX 4090	~130 FPS	1280x1280
YOLOv9 (beta)	NVIDIA A100	~190 FPS	640x640

Artefatos

- Binários darknet:
  - responsável por acessar o toolkit para efetuar treinamentos.
  - capaz de executar a inferência recebendo como parâmetros os arquivos de configuração da rede.
- `Arquivos de configuração.names yolov4-tiny.cfg yolov4-tiny.weights`

Requisitos do Tutorial

Instalar:

- Nvidia-container toolkit
- Último driver nvidia
  - extra/**cuda** 12.6.3-1 [instalado]
    NVIDIA's GPU programming toolkit
  - extra/**cuda-tools** 12.6.3-1 [instalado]
    NVIDIA's GPU programming toolkit (extra tools: nvvp, nsight)
  - extra/**cudnn** 9.5.1.17-1 [instalado]

Requisitos do Tutorial

SERPRO

Com docker:

xhost +

docker run -v /home/$USER/<DiretorioProj>/:/darknet -it --ipc=host --rm --
gp
us all --env DISPLAY=$DISPLAY -v $XAUTHORITY:/tmp/.XAuthority -e
XAUTHORITY=/tmp/.XAuthority --v
olume /tmp/.X11-unix/X0:/tmp/.X11-unix/X0
sherensberk/darknet:2204.550.1241-devel

Comandos

Treinar o detector:

./darknet detector train 'net.data' 'net.cfg' 'net.conv.14' -map -gpus 1,0

Executar o detector em uma imagem:

```
./darknet detector test net.data cfg/net.cfg
net.weights -thresh 0.25
```

Executar o detector em uma imagem imprimindo a saída:

```
./darknet detector test cfg/coco.data yolov3.cfg
yolov3.weights -ext_output dog.jpg
```

Integrando a rede em uma aplicação Python

```
# read input image
image = cv2.imread(args.image)

Width = image.shape[1]
Height = image.shape[0]
scale = 0.00392

# read class names from text file
classes = None
with open(args.classes, 'r') as f:
    classes = [line.strip() for line in f.readlines()]

# generate different colors for different classes
COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

# read pre-trained model and config file
net = cv2.dnn.readNet(args.weights, args.config)

# create input blob
blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True,
crop=False)

# set input blob for the network
net.setInput(blob)
```

Integrando a rede em uma aplicação Python

```
# function to get the output layer names
# in the architecture
def get_output_layers(net):
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
return output_layers
# function to draw bounding box on the detected object with class name
def draw_bounding_box(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
label = str(classes[class_id])
color = COLORS[class_id]
cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)
cv2.putText(img, label, (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

Integrando a rede em uma aplicação Python

```
# function to get the output layer names
# in the architecture
def get_output_layers(net):
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
return output_layers
# function to draw bounding box on the detected object with class name
def draw_bounding_box(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
label = str(classes[class_id])
color = COLORS[class_id]
cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)
cv2.putText(img, label, (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

Integrando a rede em uma aplicação Python

```
# run inference through the network
# and gather predictions from output layers
outs = net.forward(get_output_layers(net))
# initialization
class_ids = []
confidences = []
boxes = []
conf_threshold = 0.5
nms_threshold = 0.4
# for each detetion from each output layer
# get the confidence, class id, bounding box params
# and ignore weak detections (confidence < 0.5)
for out in outs:
  for detection in out:
  scores = detection[5:]
   class_id = np.argmax(scores)
   confidence = scores[class_id]
if confidence > 0.5:
  center_x = int(detection[0] * Width)
  center_y = int(detection[1] * Height)
  w = int(detection[2] * Width)
  h = int(detection[3] * Height)
  x = center_x - w / 2
  y = center_y - h / 2
class_ids.append(class_id)
confidences.append(float(confidence))
boxes.append([x, y, w, h])
```

Exemplo prático de execução

```
#libera acesso do container ao xserver
xhost +

docker run -v
/home/vfbsilva/Source/darknet_meu_serpro/:/darknet -it --
ipc=host --rm --gp
us all --env DISPLAY=$DISPLAY -v $XAUTHORITY:/tmp/.XAuthority -
e XAUTHORITY=/tmp/.XAuthority --v
olume /tmp/.X11-unix/X0:/tmp/.X11-unix/X0
sherensberk/darknet:2204.550.1241-devel
```

Como treinar sua rede

1. Organização dos Arquivos:
    1.1. Crie uma pasta para armazenar os arquivos (ex.: ~/nn/animals/).
    1.2. Copie um arquivo de configuração como modelo (ex.: yolov4-tiny.cfg) e renomeie para algo relevante (ex.: animals.cfg).

2. Crie um arquivo animals.names na mesma pasta e liste as classes (ex.: dog, cat, etc.), uma por linha, sem linhas em branco.

Como treinar sua rede

3. Crie um arquivo `animals.data` contendo:

```
classes = 4
train = /caminho/para/animals_train.txt
valid = /caminho/para/animals_valid.txt
names = /caminho/para/animals.names
backup = /caminho/para/backup
```

Como treinar sua rede

- 4. Crie uma pasta para imagens e anotações (ex.: `~/nn/animals/dataset`).
  4.1 Use ferramentas como **DarkMark** para criar arquivos de anotações no formato YOLO.
  4.2 Liste as imagens nos arquivos `train` e `valid` especificados no arquivo `.data`, com um caminho por linha.

Como treinar sua rede

5. Ajuste as configurações:
    - `5.1 batch=64` e `subdivisions` (comece com 1 e aumente se necessário).
    - `max_batches`: número de classes x 2000 (ex.: 4 classes → 8000).
    - `steps`: 80% e 90% de `max_batches` (ex.: `6400,7200`).
    - `classes`: número de classes.
    - `filters`: `(classes + 5) * 3` (ex.: 4 → `(4 + 5) * 3 = 27`).

Como treinar sua rede

6. Crie um arquivo `animals.data` contendo:

```
classes = 4
train = /caminho/para/animals_train.txt
valid = /caminho/para/animals_valid.txt
names = /caminho/para/animals.names
backup = /caminho/para/backup
```

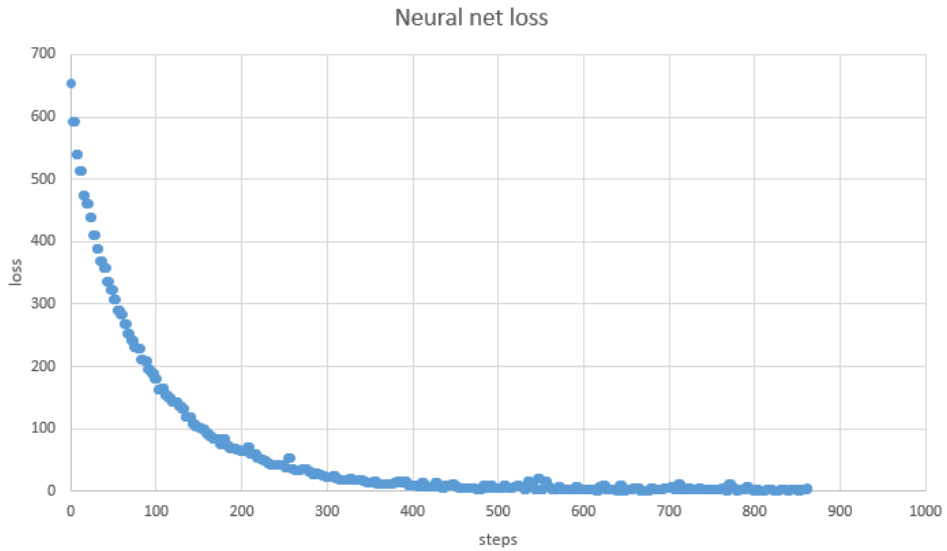
Como treinar sua rede

7. Execute o comando:

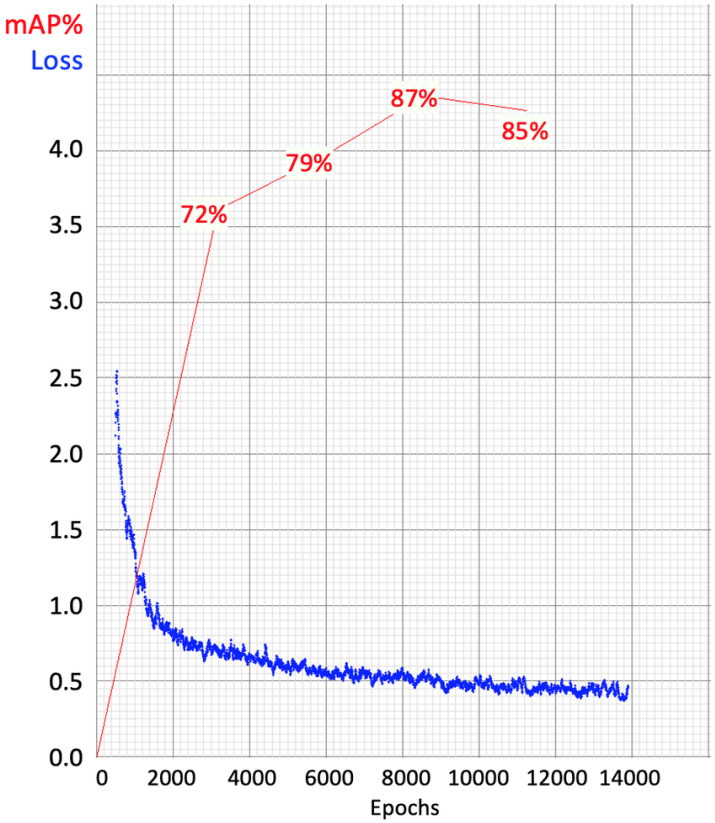
cd ~/nn/animals/

darknet detector -map -dont_show train animals.data animals.cfg

Como treinar sua rede



Como treinar sua rede



Como obter o código fonte e referências

https://github.com/hank-ai/darknet

- Paper **YOLOv7**
- Paper **Scaled-YOLOv4**
- Paper **YOLOv4**
- Paper **YOLOv3**

Obrigado!

Victor Frederico Beust da Silva
Analista

linkedin.com/in/vfbsilva

/serprobrasil
@serprobrasil
@serpro
/serpro
serpro.gov.br

SERPRO