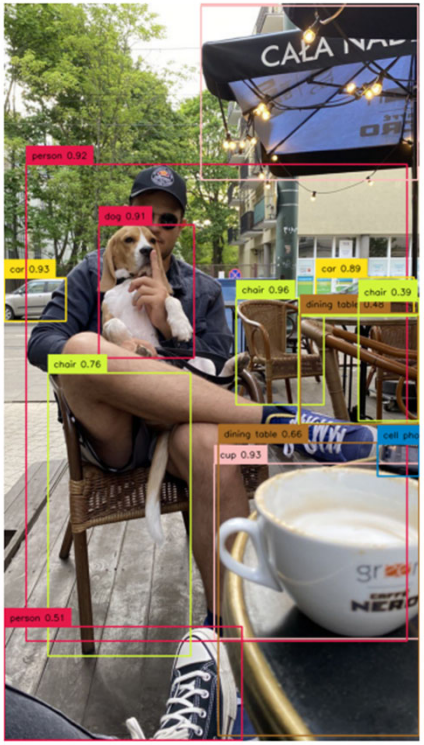
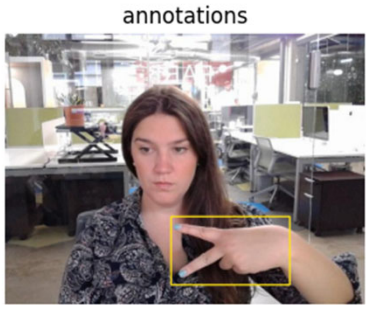
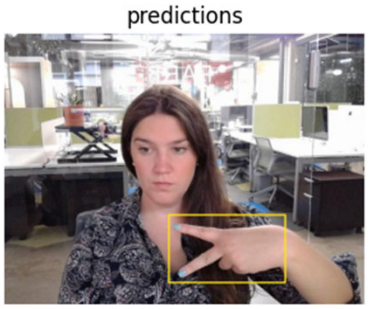
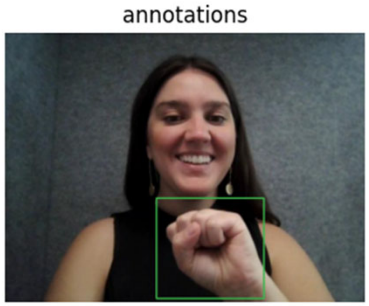
Object detection









Installing and finetuning a TF model

- Install Modell
  - Clone the tensorflow git repository & Install TensorFlow Object Detection API
  - Download Pretrained Model from Model Zoo (e.g MobileNet or EfficientNet)
- Edit Settings in Pipeline Config file
  - Specify where to find the data
  - How many target classes
  - Initial Model Checkpoint and where to save intermediate results
  - …
- Start Training with a given script `model_main_tf2.py`
- Load last Checkpoint and Export Inference Model

`Original Post:`

- https://medium.com/analytics-vidhya/training-a-model-for-custom-object-detection-tf-2-x-on-google-colab-4507f2cc6b80

Adapted Version:

- https://colab.research.google.com/drive/1uHHdTLlb9-L33p1GWriNTLrpkT8TPT7m?usp=sharing

Using the most recent model YOLO NAS

- Installation of libs
  - supergradient, roboflow, supervision
- Instanciate model with supergradient
  - models.get(architecture, numclasses, weights|checkpoint)
- Test inference
  - model.predict(picture,confidence)
- Retrive Data via roboflow library
- Finetuning
  - trainer.train(…)
- Test & Evaluate

`Original Post:`

- https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolo-nas-on-custom-dataset.ipynb

`Adapted Version:`

- https://colab.research.google.com/drive/17PgoWHRJFUAf1Sq3HZ9610irx92MBf5T?usp=sharing

Deep Learning
Lecture 10

Recap and Deepening

Prof. Dr. Rainer Herrler
Phone: 09721/ 940-8710
Email: rainer.herrler@fhws.de

Pictures from Wikipedia / Pixabay
Some Pictures generated with Stable Diffusion

Which Data Format is given in the Archives?

apples-format1.zip
- test
  - apples.tfrecord
  - apples_label_map.pbtxt
- train
  - apples.tfrecord
  - apples_label_map.pbtxt
- valid
  - apples.tfrecord
  - apples_label_map.pbtxt
- README.dataset.txt
- README.roboflow.txt

apples-format2.zip
- test
  - images
    - apple--1-_jpg.rf.7662faa1233b5fc07d695672f592c04f.jpg
    - apple--10-_jpg.rf.464143fc334f48eea9338640b7e9714a.jpg
  - labels
    - apple--1-_jpg.rf.7662faa1233b5fc07d695672f592c04f.txt
    - apple--10-_jpg.rf.464143fc334f48eea9338640b7e9714a.txt
- train
- valid
- data.yaml
- README.dataset.txt
- README.roboflow.txt

apples.format3.zip
- test
  - apple--1-_jpg.rf.7662faa1233b5fc07d695672f592c04f.jpg
  - apple--1-_jpg.rf.7662faa1233b5fc07d695672f592c04f.xml
  - apple--10-_jpg.rf.464143fc334f48eea9338640b7e9714a.jpg
  - apple--10-_jpg.rf.464143fc334f48eea9338640b7e9714a.xml
- train
- valid

Important DL websites and their main purpose

Which very useful websites for learning and exploring deep learning did you visit during this semester, and what is their main purpose:

-
-
-
-
-

Important DL websites and their main purpose

The following websites should be known by every deep learner. They are very useful for getting material and stuff for learning deep learning

- **Kaggle** – Competitions, Example Problems/Data, Example Solutions
- **Huggingface** – Hosting Models and Code, helps getting an initial iIdea about categories
- **Colab** – Hosting Jupiter Notebooks
- **Medium** – High quality articles about ML Topics,
  - Towardsdatascience – Articles / Tutorials
  - Analytics Vidhya
- **Tensorflow/Keras** – Documentation and Tutorials
- **Roboflow** – Datasets for machine learning
- …..Github, youtube …etc

Story

Suppose you are a manager in a company. Your department starts the first time with an ML-project. The developer notebooks of your department are all equipped with on-board-Graphics? What options do you have ?

-
-
-
-
-

Decision criteria for choosing the accoding system

- How complex is the learning task (Tabular data, image, video, ..)?
- Is the expected load just temporally or permanent?
- Can the hardware be shared with other departments ?
- How critical is my data, do I trust data centers ?

The cat cam

You are supposed to create a electronic cat door that only lets your cat in. Your cat is the only Siamese cat in the neighborhood. The door has a camera built in with a resolution of 255x255 pixels and is run by a raspberry pi. How would you proceed to get good results fast?

-
-
-
-
-

Cat Cam
Step 1
Find and download data

Step one – Fetching data and displaying an image

- How can we get pictures if we don’t want to shoot them?
- How many pictures will be needed ?
- Is imbalanced data a problem ?
  - Example: 1000 Siamese cats, 16000 of different breeds
  - If you train 17 cat classes and you show a dog picture, what will happen ?
- Data augmentation now or later ?

Easy fetching of data

- **Kaggle**
  - Upload json file with api-key
  - Install kaggle library
  - Use kaggle command to download

```
!pip install -q kaggle
!mkdir /root/.kaggle
!mv kaggle.json /root/.kaggle
!kaggle datasets download -d zippyz/dataset
```

- **Roboflow**
  - Visit website and copy integration code

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="XXXXXXX")
project = rf.workspace("yolov8-zz3fz").
     project("classification-de-carpe")
dataset = project.version(1).download("folder")
```

- **Github**
  - Copy link location on the website
  - Use git clone to get directory structures
  - Use linux command wget to download zip Archives

```
!git clone https://github.com/cwfid/dataset.git
!wget https://github.com/cwfid/dataset/master.zip
```

Question:

If the complete training data does not fit into memory, as it is often the case for images, what can we do ?

Streaming data from the filesystem

```python
tf.keras.utils.image_dataset_from_directory(
    directory,
    labels="inferred",
    label_mode="int",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(256, 256),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
    **kwargs
)
```

Expected Structure with label

```
main_directory/          <- Label 0
...class_a/
......a_image_1.jpg
......a_image_2.jpg      <- Label 1
...class_b/
......b_image_1.jpg
......b_image_2.jpg
```

Expected Structure without label

```
main_directory/
....a_image_1.jpg
....a_image_2.jpg
....b_image_1.jpg
....b_image_2.jpg
```

- The returned object is a so called keras.Dataset (also used by tfds)
- Images are immediately converted to tensors
- Be careful with batch_size ist the number of batches not the size of the batches
- subsets “training” and “validation” can be separated at the beginning

Further interesting stuff you can do with a dataset

- How to get the size?
  - `dataset.__len__()`
- Splitting datasets:
  - `keras.utils.split_dataset(dataset,0.8)`
- Modifying data (e.g. label modification or augmentation)
  - `dataset.map(lambda x: x)` or
  - `dataset.map(functionname)`
- Filtering data:
  - `dataset.filter(lambda x: <bool expr>)` or
- How to access items in the dataset
  - Iterating over the ds: for (img, label) in dataset: print(label)
  - Converting to a list: list(dataset)
  - dataset.take(n) gets next n elements or batches
- Many more functions: https://www.tensorflow.org/api_docs/python/tf/data/Dataset

Cat Cam
Step 2
Find Pretrained Model
Test w some Images

Finding a pretrained Model ? What do you remember?

- For **object classification**?

- For **object detection** ?

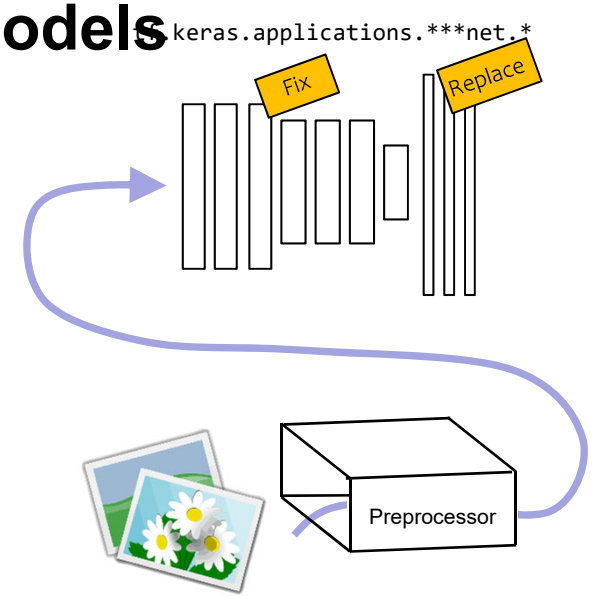
Finding a pretrained Model for Keras

- For **object classification**
  - Models directly provided in `tensorflow.keras.applications`
  - vgg16, vgg19 , resnet, inception_v3, efficientnet, efficientnet_v2
  - https://www.tensorflow.org/api_docs/python/tf/keras/applications

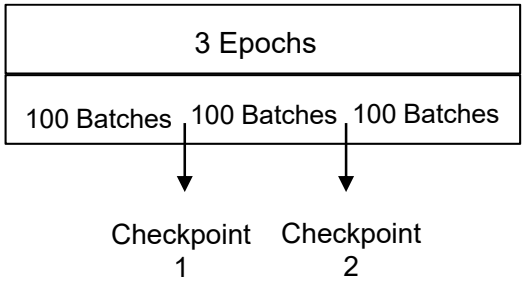
- For **object detection**
  - Download compile and install the Object Dectection API and models from https://github.com/tensorflow/models
  - Download weights from Tensorflow 2 Detection Model Zoo (github)
  - There is a newer page as well Tensorflow 2 Model Garden (github)
    Doc: https://www.tensorflow.org/tfmodels/vision/object_detection

Cat Cam
Step 3
Fine Tune Model

Fine Tuning Pretrained Classification Models



- Load model with weights and without “head”.
- Set model.trainable to false
- Rebuild head with new layers
- Check for number of output neurons in last layer
- Don’t forget to apply same preprocessing of images like the original image did!
- Check without augmentation first add augmentation later
- Add Checkpoints to store model



Documentation:
- https://keras.io/guides/transfer_learning/

Example (Lecture 8)
- https://drive.google.com/file/d/1-QgVUb6IJc4g9QU8uN26txM9EuUAXgLD/view?usp=sharing

Cat Cam
Step 4
Evaluate the model

What could we do for evaluation ?

-
-
-
-