October 29^{th}2020

Animal well-being and Monitoring

By: Shivani Deo, Pinkey Yadav, PL Manonmani,
MSc Computer Science in Machine Intelligence
IIITM-Kerala

PRESENTATION OVERVIEW

1. INTRODUCTION
2. PROBLEM STATEMENT
3. AIM OF OUR PROJECT
4. OUR APPROACH
5. ARCHITECTURES
6. RESULTS
7. CONCLUSIONS

Introduction

- In recent years, animal well-being in has become a significant concern for pet lovers consumers, farmers, and zoo. Different groups have different interpretations of animal well-being.
- For the majority of consumers, animal well-being is highly influenced by their values and experiences. Meat producers are interested in the stress animals endure because it affects meat quality.
- This creates new trend and opportunity for technology to create new development and advancement.

Problem Statement

- First to detect and classify animals.
- Second animal posture estimation.
- Third to know the animal health conditions.

Aim of Our project

- Create a program which can be installed to monitor animals and check if they are fit and alive based on their day to day activity and postures based on their movement.
- We will collect the videos of animals and send it to a server which will be saved in a cloud and then that data will be seen through an user friendly Android App whenever possible.

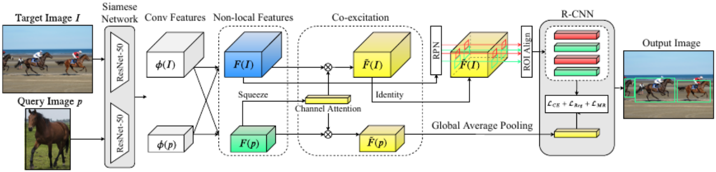
Our Approach

- The approach for this was :-
- Detecting animals
    1. trying one shot mechanism
    2. trying transfer learning mechanism
    3. Choosing the mechanism giving better results
- Detecting postures

One shot object detection

- co-attention and co-excitation Framework
- Used PASCAL-VOC and MS-COCO dataset
- Pre-trained model ResNet50 - Siamese neural network
- Non-local object proposals

One shot Architectural Figure



Architecture(One-shot Detection)

- It takes only one shot to detect multiple objects present in an image using multibox.
- It has a base VGG-16 network followed by multibox conv layers
- It's Base neural network: Extracts features
- It's Additional Conv Layers: Detect objects
- Prediction for the bounding boxes and confidence for different objects in the image is done not by one but by multiple feature maps of different sizes that represent multiple scales

Demerits of One Shot

1. **Hardware Dependency** - The model was using cuda and was dependent on additional GPU.
2. **Training Time** - Training time was more.
3. **Less Accurate** - The model gives less accurate results as we tried to search and investigate.

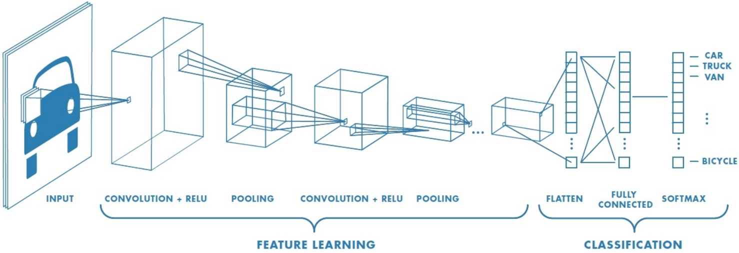
Transfer Learning and Fine Tuning

- Used VGG16
- Pre-Trained Model as Feature Extractor Preprocessor
- Image Data Augmentation
- cross-validation to train/test model
- Retrained the output dense layer

Merits of Transfer Learning

1. **More Efficient**
2. **Robust**
3. **Saves training time**
4. **Does not require lots of training data**
5. **Better performance of neural networks**

Transfer Learning Architectural Figure



Architecture(Transfer Learning)

- There are two main blocks inside of a typical CNN:-
  1. Feature Extraction.
  2. Classification.
- VGG-16 consists of 16 convolutional layers.
- Max- Pooling Layers.
- Fully connected layers.
- Output layer with Softmax activation.

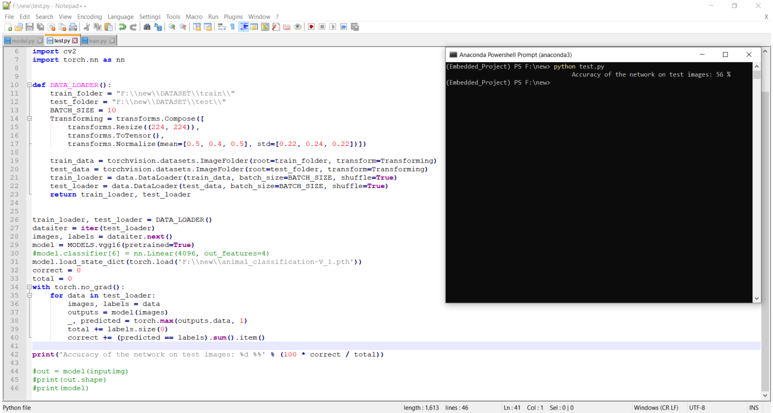
Detecting postures

- Posture tracking of animals help in getting details of their wellbeing and to get the analysis of their day to day life routine to find any abnormality.
- Approach:-
    1. We will be using ‘DeepLabCut’ for the specified purpose, DeepLabCut is a toolbox for markerless pose estimation of animals performing various tasks.

Results

1. We tried to implement one-shot detection but due to unavailblity of GPU in our systems so we were getting cuda error.
2. In order to overcome this we tried transfer learning for animal detection and classification.
3. After training the VGG-16 network we got 56% accuracy on test images.

Screenshot



F:\new\test.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
model.py
test.py
train.py
Anaconda Powershell Prompt (anaconda3)
Python file
length : 1,613 lines : 46
Ln : 41 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

```python
import cv2
import torch.nn as nn


def DATA_LOADER():
    train_folder = "F:\\new\\DATASET\\train\\"
    test_folder = "F:\\new\\DATASET\\test\\"
    BATCH_SIZE = 10
    Transforming = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.5, 0.4, 0.5], std=[0.22, 0.24, 0.22])])

    train_data = torchvision.datasets.ImageFolder(root=train_folder, transform=Transforming)
    test_data = torchvision.datasets.ImageFolder(root=test_folder, transform=Transforming)
    train_loader = data.DataLoader(train_data, batch_size=BATCH_SIZE, shuffle=True)
    test_loader = data.DataLoader(test_data, batch_size=BATCH_SIZE, shuffle=True)
    return train_loader, test_loader


train_loader, test_loader = DATA_LOADER()
dataiter = iter(test_loader)
images, labels = dataiter.next()
model = MODELS.vgg16(pretrained=True)
#model.classifier[6] = nn.Linear(4096, out_features=4)
model.load_state_dict(torch.load('F:\\new\\animal_classification-V_1.pth'))
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        images, labels = data
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print('Accuracy of the network on test images: %d %%' % (100 * correct / total))

#out = model(inputimg)
#print(out.shape)
#print(model)
```

```
(Embedded_Project) PS F:\new> python test.py
                               Accuracy of the network on test images: 56 %
(Embedded_Project) PS F:\new>
```

Conclusions

1. We are still working on the **transfer learning** to classify the animals.
2. After the classification we will be working on posture estimation using **DeepLabCut** framework.

...THANK YOU...