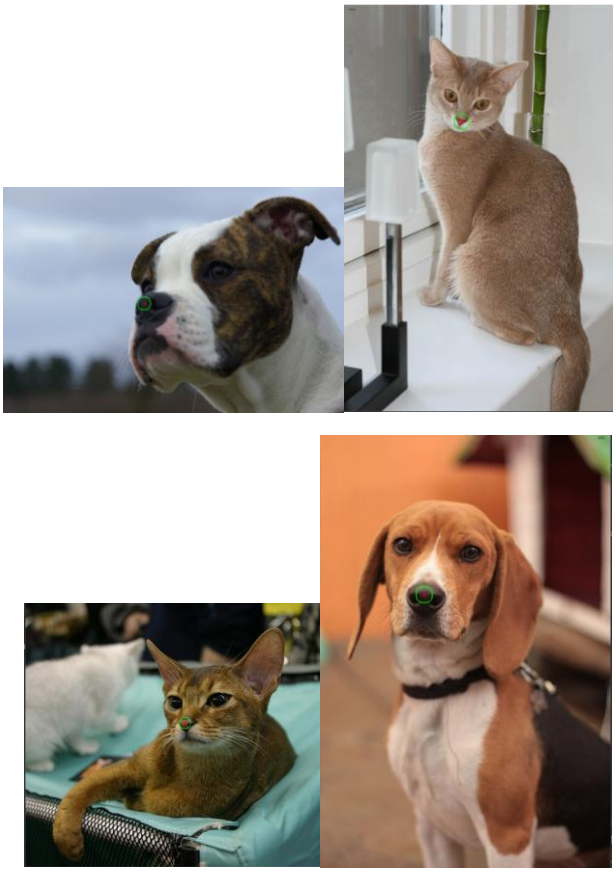
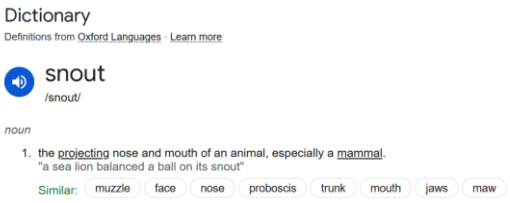


ELEC 475

Lab 2 – Pet Nose Localization

Lab 2 – Pet Nose Localization with SnoutNet



Contents

1. Introduction

The objective of this lab is to implement a *SnoutNet* model that can localize pet noses in images, and augment data to improve the performance. The dataset is ***oxford-iiit-pet-noses***, which is our reannotation of the ***oxford-iiit-pets*** dataset. You should be able to reuse elements of the Lab1 code base in pursuit of this goal. You may use any large language model (e.g., ChatGPT, Claude) for this assignment. When prompting, be explicit and precise about what you want. Always test and fact-check every piece of code the model generates — do not accept code outputs at face value. Verify imports, function names, arguments, and tensor shapes, etc. - and of course make sure that it runs correctly!

2. Task

Your task is as follows:

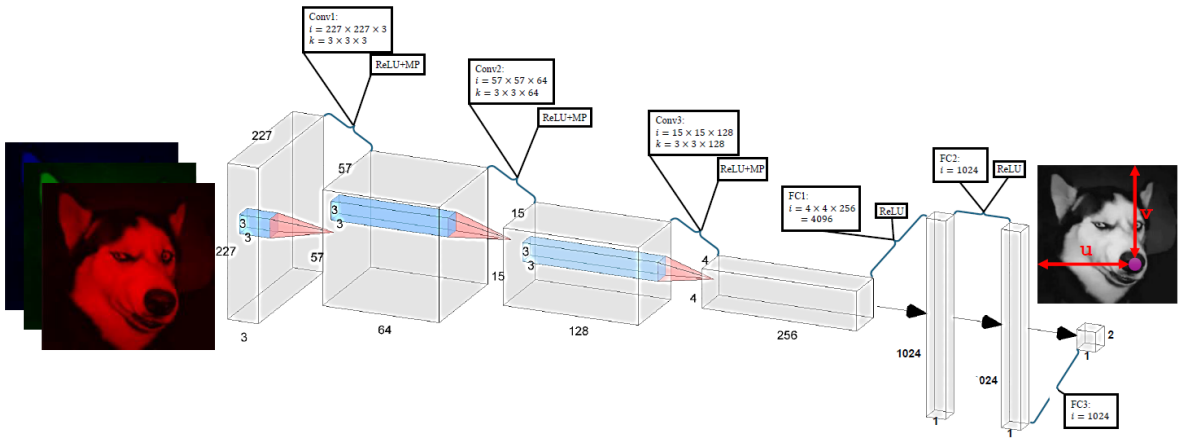


Figure 1 – SnoutNet Architecture Diagram

2.1 Step One – Model

Develop a model to regress (i.e. estimate) the pet nose locations in each image in the dataset, based on the diagram of SnoutNet provided in Figure 1. Code this into a separate 'model.py' source file. Keep in mind that you will need to follow the exact configuration of the architecture, as shown in Figure 1. The PyTorch official documentation contains all of the information for you to build SnoutNet, e.g. Conv2d — PyTorch 2.4 documentation, MaxPool2d — PyTorch 2.4 documentation . You will also need to reshape the feature map after convolution layers to feed it into fully connected (linear) layers (torch.Tensor.view — PyTorch 2.4 documentation).

After building your model, write a simple script and prepare a simple random dummy input tensor, with the shape of BxCxWxH = 1 x 3 x 227 x 227 (= batch_size x rgb_channels x img_width x img_height), and forward it into the model you created. If your network architecture has been coded correctly, then an output tensor of shape 1 x 2 (= batch_size x uv_coor_dim) should be expected.

Hint: Try feeding the model architecture image above into an LLM, to generate the initial model.py module.

2.2 Step Two – Data

Download the data: Download the ***oxford-iiit-pets*** dataset with reannotation from the ELEC 475 Google Drive into your local drive. (You should ***download only*,** please do not modify anything within the Google Drive). Alternatively, you can copy the dataset into your own Google Drive, and make it visible in your Google Colab (Snippets: Accessing files - Colab (google.com)). The dataset file structure is as follows:

- images-original

    -- images # actual images

        --- Abyssinian_1.jpg

        --- Abyssinian_2.jpg

        ……..

    -- images.tar.gz # original ***oxford-iiit-pets data***

- test-noses.txt #annotation of our test set

- train-noses.txt #annotation of our train set

Develop a Custom Dataset: Your custom Dataset class (Datasets & DataLoaders — PyTorch Tutorials 2.4.0+cu121 documentation) should be able to accommodate different types of data augmentation (Transforming and augmenting images — Torchvision 0.19 documentation (pytorch.org)). Your Dataset should initialize the path to images (images-original/images/), and groundtruth (GT) labels (train-noses.txt and test-noses.txt). The GT labels look like the following:

```
beagle_145.jpg,"(198, 304)"
shiba_inu_136.jpg,"(182, 203)"
english_cocker_spaniel_181.jpg,"(145, 293)"
english_cocker_spaniel_17.jpg,"(126, 206)"
chihuahua_165.jpg,"(122, 122)"
…
```

On each line, the string before the comma is the image's file name, and the quoted ordered pair after the comma is the uv (i.e. (x,y)) pixel coordinates of the nose. Be mindful with these coordinates, as different libraries handle these images differently, and the sequence of u and v coordinates can sometimes be flipped. You will also need to reshape the images as the input images to SnoutNet should always be of size 3 x 227 x 227.

Reality Check your DataLoader: Once you have developed your Dataset and DataLoader, write a quick reality check routine that iterates through all images and labels, prints the values, and (optionally, selectively) visualizes the images and the GT labels. This is crucial, as incorrect data loading can lead to unpredictable behavior of the network.

2.3 Step Three – Training

Write a training script and train your SnoutNet model, using the train partition of the dataset. For this step, don't augment the data. Use the test partition for validation purposes (i.e. just to track the progress of the loss plot, and not as part of network parameter optimization). Keep in mind that only regression losses can be used as we are handling a regression task, letting the model estimate the location of pets' snouts (as uv-coordinates within the image frame).

2.4 Step Four – Testing

Test your trained model on the dataset test partition. Calculate the localization accuracy statistics, i.e. the minimum, mean, maximum, and standard deviation of the Euclidean distance from your estimated pet nose locations, to the ground truth pet nose locations.

2.5 Step Five – Data Augmentation Training and Testing

Repeat Steps Four and Five using at least two different types of data augmentation. Include these as options in your training scripts.

2.6 Step Six – Finetune Pretrained Models

Extend the existing codebase, by implementing two separate model variants: one with an AlexNet backbone and one with a VGG16 backbone, each loaded and initialized with pretrained torchvision weights. Since these architectures were originally designed for ImageNet classification, adapt them for our regression task by replacing the final classification layers with layers suitable for predicting continuous values. Train and evaluate each variant independently—AlexNet-only, then VGG16-only—under two conditions: (1) without data augmentation and (2) with data augmentation. Do not combine the backbones or share weights between runs; all other training and evaluation settings should remain the same for fair comparison.

2.7 Step Seven – Ensemble Model

From the three models that you've implemented, create an ensemble model that combines the results from all three to produce a new (hopefully better) result.

3. Deliverables

The deliverables as described below comprise the following:

- Your completed code
- Your report.

3.1 Completed Code

In addition to submitting the code, **each model** must come with its own **text file** containing the commands to run it from the PyCharm terminal.

Each text file should list **four commands**, with each command doing one of the following:

- **Train without Augmentations**
- **Train with both Augmentations**

- **Test**
- **Visualize**

Name the files according to the model (SnoutNet-A and SnoutNet-V denote the AlexNet-based and VGG-based variants of SnoutNet, respectively):

- **SnoutNet.txt**
- **SnoutNet-A.txt**
- **SnoutNet-V.txt**
- **SnoutNet-Ensemble.txt**

Hint: The first 3 files will have 4 lines each — one per command. The last file will have 2 lines for the final two commands (test and visualize), as the ensemble model will use the pretrained versions of the previous three models.

Additionally, upload your trained models to an external drive (One drive, Google drive, etc,.) and share the link in a text file named "Weights_link.txt". Make sure that access is granted to the individual who needs to access the link (i.e. your TA).

3.2 Report

You report should be relatively brief (4-6 pages), and include the following information:

1. What is your network architecture?
   You should describe the SnoutNet structure in sufficient detail so that it can be reimplemented solely from the textual description.

2. How did you implement your custom Dataset?
   You should describe how the images and labels are loaded, and how you conduct a quick fact check on the Dataset and DataLoader you developed.

3. Describe all hyperparameters used in training, and describe the hardware used. How long did the training take? Include the loss plot.

4. Describe the different data augmentation methods you implemented. For each method, explain whether the labels needed to be adjusted or whether they could remain unchanged.

5. Describe your experiments. Include a description of the hardware that you used, and the time performance for training for each augmentation variation, and for testing (e.g. msec per image). Describe the localization accuracy statistics, as an error measure. Include a table that shows an ablation study of the augmentation methods, and how they impacted the quantitative results, e.g.:

Model	Augmentation	Localization Error				Localization Error (4 Best)				Localization Error (4 Worst)			
		min	max	mean	stdev	min	max	mean	stdev	min	max	mean	stdev
SnoutNet													
	x												
SnoutNet-A													
	x												
SnoutNet-V													
	x												
SnoutNet-Ensemble													
	x												

For the columns labeled "4 Best" and "4 Worst", report the statistics corresponding to the four best and four worst predictions across the test set.

Additionally, include four visualized examples for each model (SnoutNet, SnoutNet-A, and snoutNet_v). Use trained models with both augmentations for visualizations.

6. Discuss the performance of your system. How well did it perform? Was the performance as expected? Discuss which (if any) of the augmentation methods were beneficial. What challenges did you experience, and how did you overcome them?

7. Describe how your ensemble approach combined the information from the three models. Discuss how this impacted performance.

8. Share the link to your conversation with LLM of your choice (ChatGPT, Claude, etc.). How did you make sure that code or concepts generated by LLM were not hallucinations?

4. Submission

The submission should include all of your source code, and the report described in Section 3.

All deliverables should be compressed into a single **<zzz>.zip** file, where filename **<zzz>** is replaced with your student number. If you are in a team of two, then concatenate both student numbers, separated by an underscore. The zipped directory should include your code and scripts, your trained parameter files, and all output plots and images.

Do not include the Google Drive data that you downloaded in Step 2.2 in your zip file.

The report should include a title (e.g. minimally ELEC 475 Lab 2), your name and student number. If you are working in a team of two, then include the information for both partners in the report (but only make one submission in OnQ).

For each model, include a text file containing four separate commands (Listed is Section 3.1). Ensure that your code can be executed successfully using these provided commands without additional modifications.

The marking rubric is as follows:

Item	mark
SnoutNet Train/Eval/Vis	2
SnoutNet-A Train/Eval/Vis	1
SnoutNet-V Train/Eval/Vis	1
SnoutNet-Ensemble Eval/Vis	1
Report	2.5
Correct submission format	0.5
Total:	**8**