ELEC 475 Lab 2 Report

SnoutNet

Torin Lam: 20269270
Daniel Duinko: 20229482

SMITH ENGINEERING
Queen's University

Network Architecture:

The SnoutNet architecture is a convolutional neural network designed to perform regression on 227×227 RGB images, predicting the two-dimensional pixel coordinates of a pet’s nose. The network begins with an input tensor of size (B x 3 x 227 x 227) where B is the batch size and the three channels correspond to the RGB color space. The first convolutional layer applies 64 filters of size 3×3 with a stride of 1 and padding of 1, preserving the spatial dimensions. A ReLU activation follows this convolution, and the feature map is then down sampled using a 3×3 max-pooling operation with a stride of 4, reducing the feature map to 57×57x64.

The second convolutional layer increases the channel depth from 64 to 128 using 3×3 filters with a stride of 1 and padding of 2. This layer is followed by another ReLU activation and a 3×3 max-pooling layer with a stride of 4, which reduces the spatial size from 57×57x64 to 15×15x128. The third and final convolutional layer expands the channel depth to 256 using 3×3 filters with a stride of 1 and padding of 1. After another ReLU activation, a final max-pooling layer with kernel size 3 and stride 4 is applied, producing a compact 4×4 x256 feature map.

The resulting feature maps are flattened into a 4096-element feature vector for each image, which is passed through a series of fully connected layers for regression. The first fully connected layer maps the 4096 features to 1024 neurons, followed by a ReLU activation. A second fully connected layer maintains this dimensionality with another 1024 neurons and a ReLU activation. Finally, the output layer is a fully connected layer with two neurons, representing the predicted ((u, v)) pixel coordinates of the pet’s nose. No activation function is applied after this final layer, as the network performs direct regression instead of classification.

Overall, SnoutNet can be summarized as a three-block convolutional feature extractor followed by a three-layer fully connected regressor. ReLU activations are applied after each hidden layer, and the model is trained end-to-end using mean squared error (MSE) loss between the predicted and ground-truth nose coordinates in pixel space. This design allows the network to learn hierarchical spatial features from raw images while regressing directly to numerical coordinate outputs.

Custom Dataset Implementation:

The custom Dataset class that was developed takes the path to the image directory and an annotation file (train-noses.txt/ test-noses.txt). It reads the annotation file line by line, splitting each line at the first comma to separate the image filename from the nose coordinates. The coordinate string, which appears in the form "(x, y)", is then converted into a numerical tuple and paired with the full image path. This information is stored internally as a list of image and label pairs that can be accessed later.

When the dataset is queried, the class loads the image in RGB format using the Python Imaging Library (PIL), records its original width and height, and then resizes it to a consistent 227 × 227 pixels, the input size expected by SnoutNet. Due to resizing changing the scale of the image, the ground truth coordinates are proportionally adjusted so that the nose location remains correct in the resized image. The image is then transformed into a tensor that PyTorch can process, and both the tensor and its corresponding label are returned to the DataLoader during training or testing.

A reality check was performed to verify that the Dataset and DataLoader were working properly. This involved iterating through several samples, printing the shapes of the image tensors and labels to confirm they matched expectations (3 × 227 × 227 for images and 2 for labels). Additionally, a few images were visualized using Matplotlib with the ground truth nose coordinates plotted as red points. The red points correctly appeared on the pets noses, confirming that the data was being read, scaled, and paired correctly before being passed into the neural network.

Training Hyperparameters:

Table 1: Table of hyperparameters and hardware used for this lab

Hyper Parameters:	**Learning Rate**	**Batch Size**	**Optimizer**	**Epochs**	**Loss Functions**
	1e-4	64	**Adam:** with a weight decay = 1e-5	20	Mean Squared Error
Hardware:	**Computer**	**GPU**	**RAM**		
	Macbook M4 Pro	M4 Pro 16 Core	24 GB		

Total Training Time:	**Custom**	**AlexNet**	**VGG16**		
Non-Augmented	250.2 s	320.8s	2434.4		
Augmented	336s	272	2514.2		

Training Times:

Snoutnet no Aug: Epoch: 12.51s, (Avg: 2.67 ms/image)

Snoutnet with Aug:Epoch Time: 16.80s (Avg: 2.71 ms/image)

Snoutnet-A with no Aug: Epoch Time: 16.04s (Avg: 2.59 ms/image)

Snoutnet-A with Aug: Epoch Time: 13.60s (Avg: 2.20 ms/image)

Snoutnet-V no Aug: Epoch Time: 121.72s (Avg: 19.66 ms/image)

Snoutnet-V no Aug: Epoch Time: 125.71s (Avg: 20.30 ms/image

Loss Plots

Please see the **appendix** for all the **loss plots.**

Data Augmentation:

To improve the robustness of the model and help it generalize to new images, two types of data augmentation were implemented: horizontal flipping and random erasing.

The horizontal flip randomly mirrors an image from left to right with a 50% probability. Because this operation changes the spatial orientation of the image, the ground-truth nose coordinates must also be adjusted. Specifically, the x-coordinate of the nose is recalculated as x'=(w-1)-x where w is the image width and x are the original horizontal position. This ensures that when the image is flipped, the ground truth label points to the correct coordinates.

Initially, colour jittering was chosen as an augmentation but quickly changes to random erasing because it provides a more meaningful challenge for the network by simulating real-world situations such as parts of the snout being hidden by fur, shadows, or other objects, rather than simply altering image brightness or color. The random erasing augmentation randomly selects a small rectangular region in the image and replaces it with a constant value (such as black or the dataset mean). This encourages the model to learn stronger spatial and contextual features for nose localization, improving its robustness to partial visibility rather than variations in lighting alone. Since this transformation does not move or distort the image geometry, the ground-truth nose coordinates remain unchanged.

Together, these two augmentations increase the diversity of the training data and help the model generalize better by making it less sensitive to image orientation or partial occlusion of the pet's nose.

Experiments:

Error Measurements

Table 2: Table containing the error measurements for the overall data, 4 best and 4 worst of the SnoutNet, SnoutNet-A, SnoutNet-V and Ensemble models

Model	Augmentation	Localization Error				Localization Error (4 Best)				Localization Error (4 Worst)			
		min	max	mean	stdev	min	max	mean	stdev	min	max	mean	Stev
SnoutNet		0.7	133.13	34.43	22.32	0.67	1.84	1.41	0.43	109.33	133.13	119.53	10.17
	x	0.7481	145.57	32.33	22.10	0.75	1.51	1.08	0.33	102.87	145.58	120.97	17.89
SnoutNet-A		2.7	160.60	39.27	18.66	2.7	4.19	3.47	0.63	105.68	160.6	125.07	21.03
	x	3.66	161.33	33.37	16.75	3.66	4.20	4.04	0.22	109.50	161.13	125.08	21.22
SnoutNet-V		0.47	94.30	8.93	10.30	0.47	0.64	0.54	0.06	72.18	94.30	80.04	8.93
	x	0.41	80.65	9.09	8.68	0.41	0.56	0.49	0.05	58.60	80.65	69.21	8.11
Snounet-Ensemble		0.83	118.33	20.63	14.85	0.83	1.28	0.98	0.18	86.17	118.33	101.06	11.54
	x	0.97	114.93	18.55	13.37	0.97	1.25	1.10	0.10	88.76	114.44	99.2	9.41

Visualization

Please see the pictures in the **results folder**, found in the **final zip file**. In the folder, there are the 4 best, and 4 worst visualizations for each model, both augmented and non-augmented.

Performance Discussion:

The system's performance was generally as expected. All models successfully learned to localize the pet's nose with reasonable accuracy, showing decreasing training and validation losses over time. The custom SnoutNet model performed well but showed some overfitting without augmentation, which improved after applying horizontal flipping and random erasing. The AlexNet and VGG versions performed more consistently, with VGG achieving the lowest overall error as expected due to its deeper architecture and stronger feature extraction capability. Overall, the results aligned with expectations, data augmentation improved robustness, and larger, pretrained-inspired architectures delivered better localization accuracy.

The group used two augmentation methods: horizontal flipping and random erasing. Results from the loss plots and error statistics table show that these augmentations provided measurable benefits, especially for the custom SnoutNet and VGG16-based models. For SnoutNet Custom, the augmented version achieved a lower mean localization error and smoother validation loss curve, indicating better generalization and reduced overfitting. Horizontal flipping helped the model learn facial symmetry, while random erasing improved robustness to occlusions and missing features. For SnoutNet-AlexNet, augmentation had little effect since the model already generalized well due to its larger receptive fields. In contrast, SnoutNet-VGG16 showed the most consistent improvement, with lower mean error and smaller variation between training and validation losses. Overall, the combined flip and random erasing strategy improved robustness to mirrored orientations and partial occlusions, resulting in more stable training and better performance on unseen images.

Challenge 1: Inputting partial instructions

You found that inputting instructions for training, testing, and visualizing one at a time, rather than as a complete set, created discrepancies between the different files. This piecemeal approach meant you had to later consolidate all the files into a single prompt to resolve the differences. This also complicated the submission process, as the commands needed to run the functions had to be updated several times. The key lesson was that providing all related instructions in one comprehensive prompt is far more efficient and ensures consistency from the start.

Challenge 2: File and directory management

Initially, all your plots were being saved in the same root folder as the Python scripts and text files. This cluttered the workspace and made it difficult to find the specific files you needed for editing. You learned that a clean directory structure is essential, and you had to use additional prompts just to move all the visualization plots into their own dedicated folders. The takeaway is that establishing an organized folder structure should be part of the initial project setup.

Challenge 3: Short prompts vs. detailed prompts

You experimented with both short, simple prompts and long, detailed prompts and found a significant difference. While short prompts are faster to write, they often result in intermediate or incomplete answers that require more follow-up prompts. You learned that in almost any situation, longer and more detailed prompts saved more time in the long run, even with the initial effort required to write them. By providing the model with more context, it was able to output more "final" answers rather than intermediate ones, reducing the overall back-and-forth.

Ensemble Approach:

The ensemble combines predictions from the custom, AlexNet, and VGG16 models using simple averaging. For each test image, the script first obtains the (x, y) snout coordinates predicted by each of the three models individually. It then calculates the arithmetic mean of these three coordinate pairs to produce the final ensemble prediction. This averaged coordinate is subsequently compared against the ground truth to evaluate the ensemble's performance.

This averaging approach typically improves performance and robustness. By blending the outputs, the ensemble smooths out uncorrelated errors from the individual models, often resulting in a lower mean error and reduced error variance compared to any single model. It leverages the architectural diversity of the networks (custom, AlexNet, VGG16), combining their differently learned features. While not guaranteed to outperform the single best model, this method generally provides more stable and reliable predictions by mitigating the impact of outliers or poor performance from any one model.

LLM Usage:

Link 1: https://gemini.google.com/share/eb8050e23b4e

Link 2: https://gemini.google.com/share/280e33d00a8a

Explain how you verified that the code or concepts from the LLM were correct and not "hallucinations.":

The model that was used was Gemini. There are a few methods that were used to ensure that hallucinations did not occur:

1. **Explicit Instructions:**
   - Provided detailed explanations for code functionality.
   - Included specific instructions like “Do this, however, don’t remove this or don’t do this.”
   - Emphasized leaving certain code sections unchanged while modifying others.

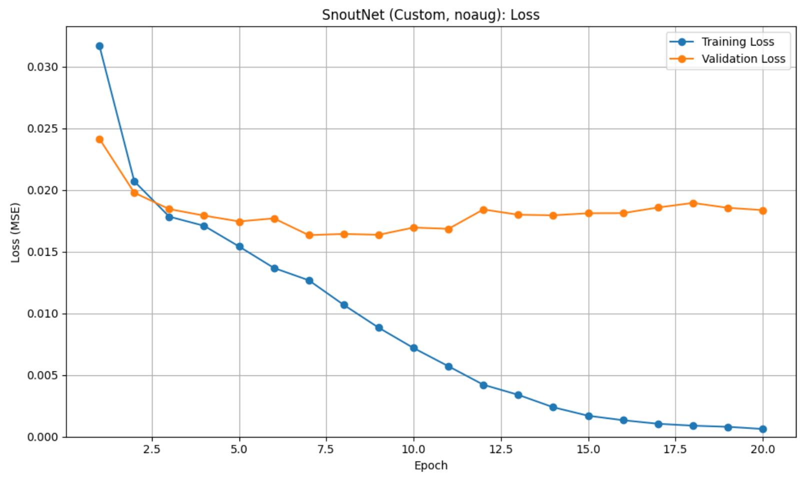
2. **File Specification:**
   - Clearly identified which files needed to be edited rather than allowing the model to guess.

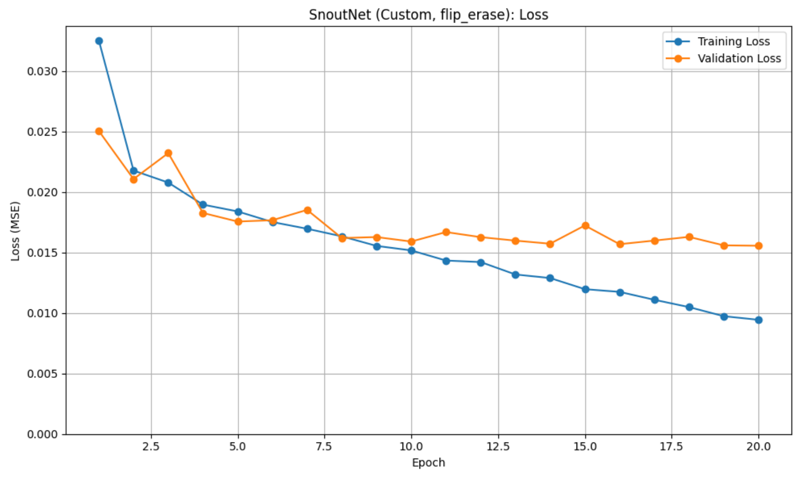
3. **Gemini’s Canvas Feature:**
   - Utilized Gemini’s canvas option to display code on the right side of commands.
   - Modified specific lines of code by highlighting the code in canvas and running a sub command within to modify only the chosen lines.
   - Allowed for reading and reviewing code as it was generated, facilitating line-by-line changes.

Appendix

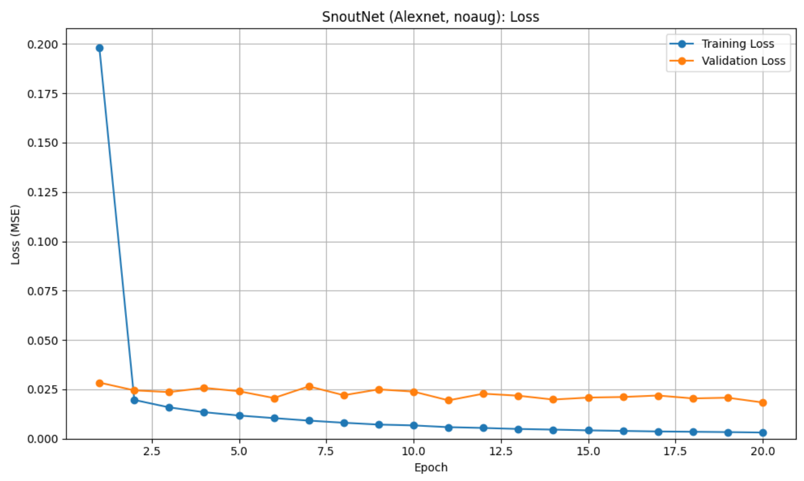
Loss Plots:

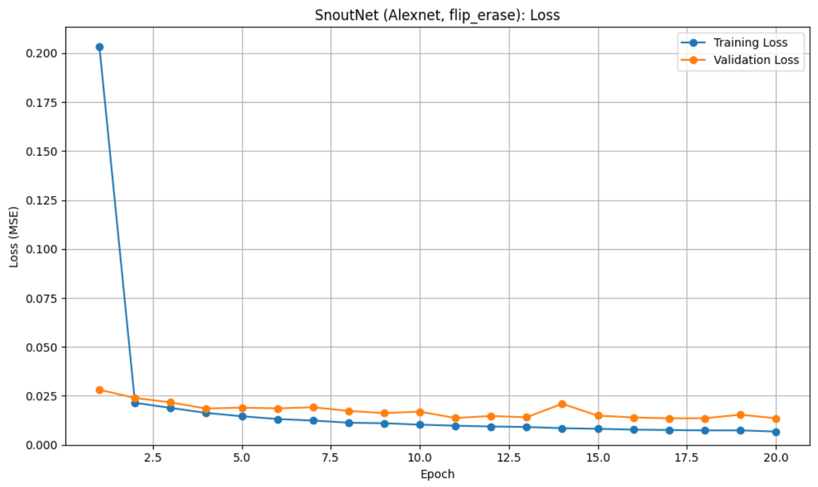
SnoutNet Custom



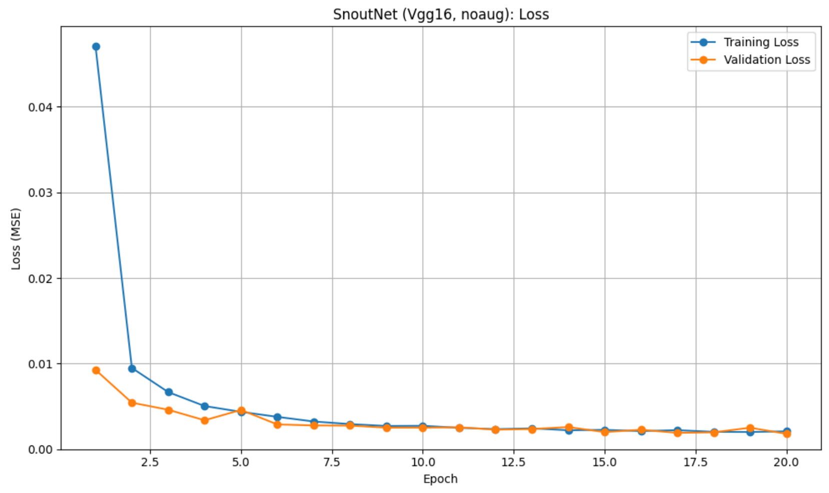


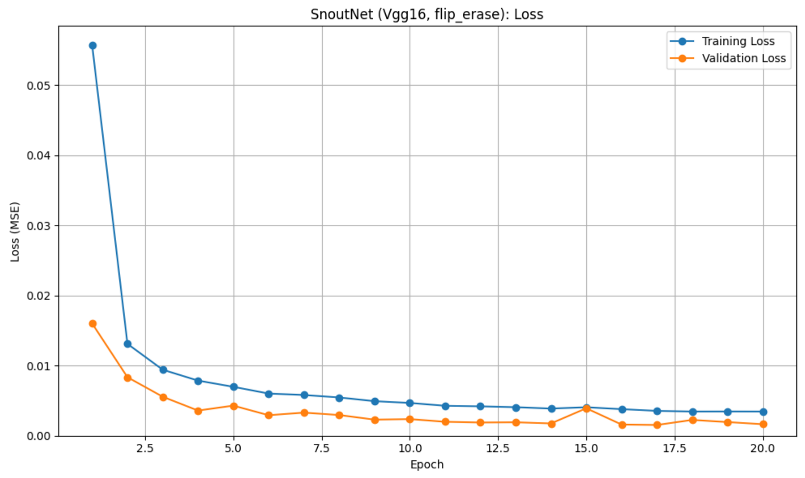
SnoutNet AlexNet





SnoutNet VGG16


SnoutNet (Vgg16, noaug): Loss
Training Loss
Validation Loss
Loss (MSE)
Epoch


SnoutNet (Vgg16, flip_erase): Loss
Training Loss
Validation Loss
Loss (MSE)
Epoch