

ELEC 475

Tahira Hossain

20230310

2023-12-05

Lab 5: Pet Nose Localization

Step 2

In the development of the pet nose localization model, the pre-existing ResNet-18 model was used. This decision was driven by the need for a robust and efficient solution capable of accurately identifying and localizing pet noses in images. ResNet is known for skip connections that allow the network to learn an identity function, ensuring that deeper layers can perform at least as well as shallower ones. ResNet-18 has 18 layers.

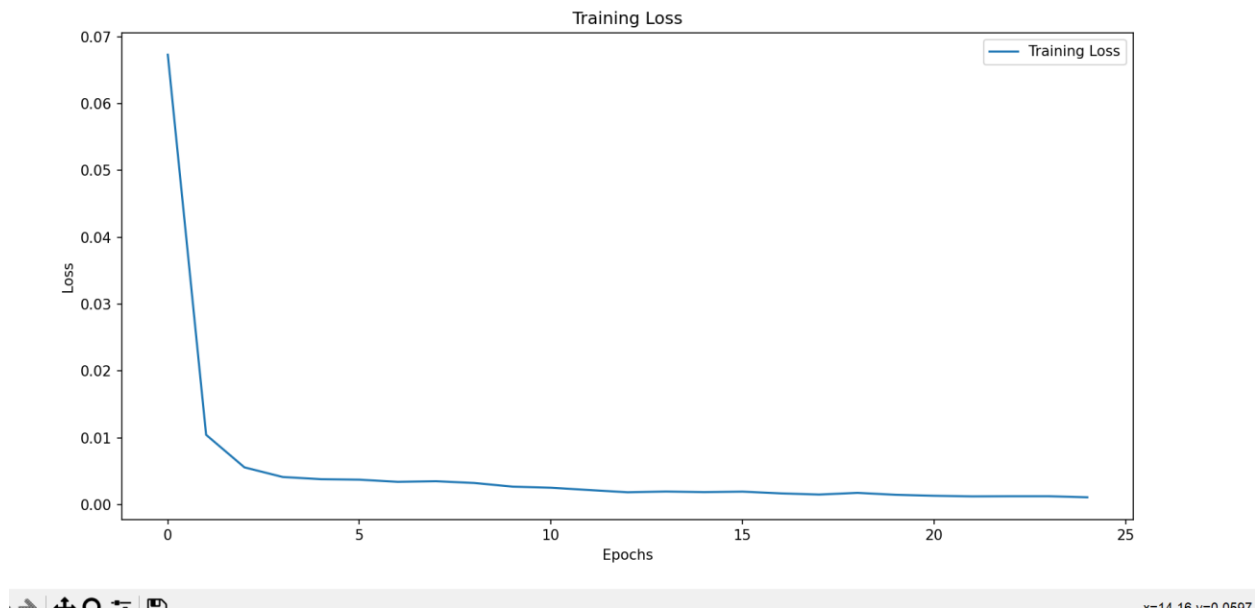
ResNet architectures, with their residual connections, effectively address the vanishing gradient problem, allowing deeper models to learn efficiently. The model is complex enough to learn detailed features but not so deep that it becomes computationally expensive. ResNet-18 comes with weights pre-trained on the ImageNet dataset. This transfer learning approach can boost performance, especially since the dataset is not extremely large.

The original ResNet-18 is designed for classification tasks with 1000 classes (ImageNet). For our task, we needed to predict two continuous values (x, y coordinates of the nose). Therefore, the final fully connected layer of ResNet-18 was modified to output two values instead of 1000. Specifically, with the assignment statement `model.fc = nn.Linear(num_fts, 2)` where `num_fts` is the number of features coming into the final layer.

Input images were resized to 224x224 pixels to match the input size expected by ResNet-18. Standard normalization using mean and standard deviation values based on the ImageNet dataset were applied. The existing resnet18 model from PyTorch Models was used for this project. The original paper referenced is Deep Residual Learning for Image Recognition by He et al., 2015.

Step 2

The loss plot generated by the training algorithm is shown in the figure below.



The hyperparameters used during training are as follows.

1. Learning Rate: 0.0001

This is the step size used for each iteration of the weight update in the training process. A smaller learning rate was chosen to ensure more precise updates and prevent overfitting.

2. Batch Size: 32

This refers to the number of training samples used in one iteration. A batch size of 32 is used for a balance between the efficiency of larger batches and the more frequent updates of smaller batches.

3. Number of Epochs: 25

The model was set to train for 25 epochs, which was determined to be sufficient for the model to learn effectively without overfitting.

4. Loss Function: Mean Squared Error

Mean Squared Error Loss calculates the square of the difference between the predicted and actual values. It is very suitable for regression problems like localization.

5. Optimizer: Adam

Adam optimizer was used, known for its effectiveness in handling sparse gradients and adapting the learning rate during training.

6. Learning Rate Scheduler: ReduceLROnPlateau

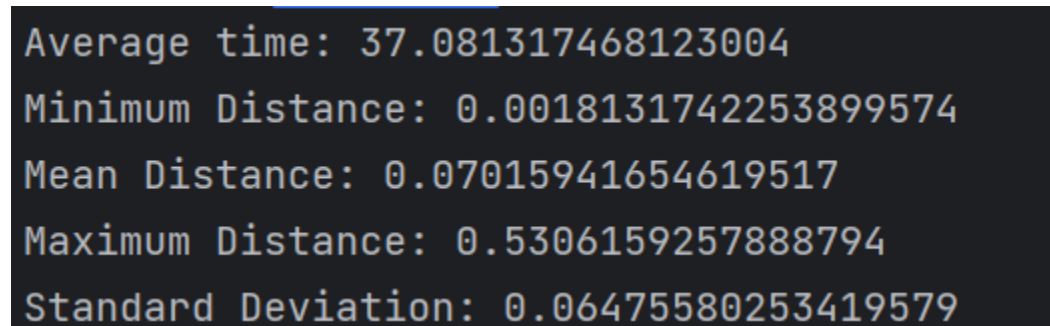
This scheduler reduces the learning rate when a metric has stopped improving, which is useful for fine-tuning the model in the later stages of training. This prevents overfitting.

Hardware Used

The model training was set to utilize a CUDA-capable GPU, if available. This significantly speeds up the training process by allowing for parallel processing of large batches of data. In the absence of a CUDA-capable GPU, the model defaults to using the CPU.

During training, the CPU was used. This took around 3 hours to complete.

Step 4



```
Average time: 37.081317468123004
Minimum Distance: 0.0018131742253899574
Mean Distance: 0.07015941654619517
Maximum Distance: 0.5306159257888794
Standard Deviation: 0.06475580253419579
```

The localization accuracy statistics shown in the image indicate the following:

Minimum Distance: The closest predicted nose location to a ground truth location was approximately 0.0018 normalized units. This is the smallest error recorded among all the test samples meaning that in the best-case scenario, the model's prediction was nearly perfect. Such a low value suggests that the model has the potential to make highly accurate predictions.

Mean Distance: On average, the predicted nose locations were 0.0702 normalized units away from the ground truth locations.

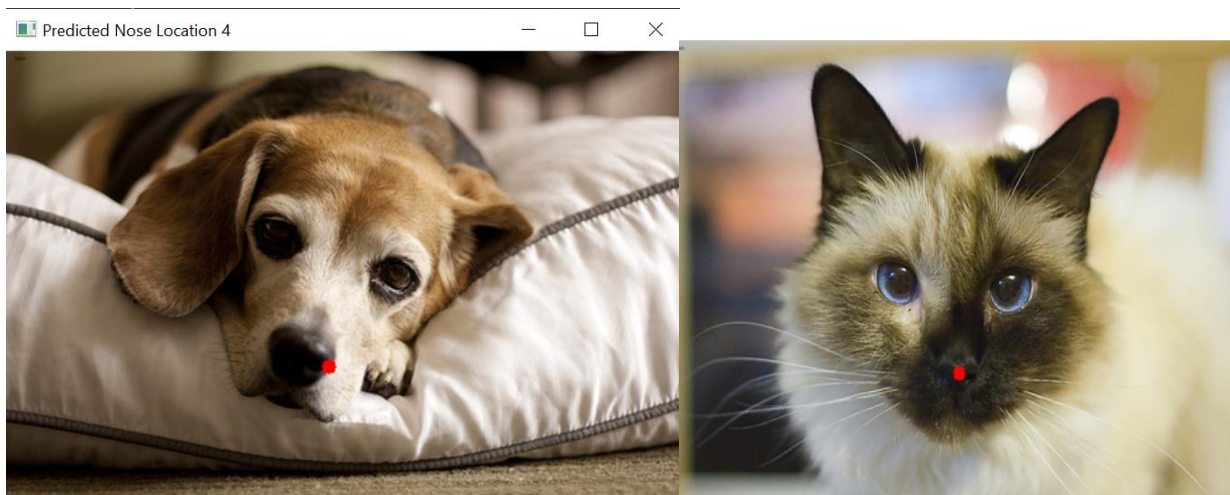
Maximum Distance: The furthest predicted nose location was 0.5306 normalized units from the ground truth, indicating that there were some predictions with significant error. It's an indication of the worst-

case scenario where the model's prediction was far off from the ground truth. This can be due to certain outliers such as the photo shown below.



Standard Deviation: The standard deviation of 0.0648 normalized units shows that there was some variability in the prediction accuracy, but not excessively.

In summary, the method worked well and provided fairly accurate predictions. The worst-case scenario is not too extreme and on average, all the predictions are close to the actual pet nose locations if not exactly there. The images below display the test results with the red dot being the predicted location.



A CPU was used during testing and an average time of 37.08 msec per image was taken by the program.

Discussion: Discuss the performance of your system. Was it as expected? What challenges did you experience, and how did you overcome

Discussion:

One of the problems encountered was resizing of the labels and images to fit the ResNet architecture requirements. The image was resized using transforms in Python, however the sizes of the label coordinates were left as is. This caused large loss values. It was fixed by dividing the size of the coordinates by the corresponding image width and height. For visualization during testing, these had to be reverted back to their original values to accurately depict the predictions on the original images. The learning rate at 0.001 also caused higher training losses than expected so 0.0001 had to be chosen. A few images in the dataset came corrupted at first and had to be deleted.

Once training and testing were completed, the results were as expected. Outliers where the predictions are not on the nose exist but are in the minority of the test cases. The speed of the training process was faster than expected considering a CPU was used.