ELEC 475

Lab 5: Pet Nose Localization

Jessica Sider (20232610)

Tomi Kofo-Alada (20152073)

Step 2

In this project, we employed a modified version of the ResNet-18 model, specifically adapted for the task of pet nose keypoint detection (localization). ResNet-18, originally introduced in the paper "Deep Residual Learning for Image Recognition" by K. He, X. Zhang, S. Ren, and J. Sun (2015), is a widely acclaimed convolutional neural network known for its depth and efficiency, primarily used for image classification tasks. The model can be found in the PyTorch “torchvision” library, which provides pre-trained weights for various applications. We downloaded and utilized the pretrained version. The choice of ResNet-18 was driven by its balance between complexity and performance. The model's residual learning framework eases the training of deeper networks by addressing the vanishing gradient problem. This characteristic is particularly beneficial for our application, which requires the model to learn intricate patterns in images for accurate keypoint detection.

The original ResNet-18 is designed for classification tasks, outputting a vector of class probabilities. To tailor it for localization, a regression task, we made the following modifications:

- We retained most of the ResNet-18 architecture, using its layers for feature extraction. This is done by keeping all layers of the original ResNet-18 up to the second-to-last layer (excluding the final classification layer).
- After feature extraction, we incorporated an adaptive average pooling layer. This layer dynamically adjusts its pooling size, ensuring a fixed-size output regardless of the input image size, which is crucial for consistent feature mapping.
- The key modification is the replacement of the original fully connected layer with a new regression head. This head comprises a linear layer with an output size of 2, corresponding to the (x, y) coordinates of the pet nose in the image.

The first part of the model (features) consists of the convolutional and residual blocks of the original ResNet-18, providing a feature extraction mechanism. Following the feature extraction layers, an adaptive average pooling layer (avgpool) standardizes the output size. The final part is a regression head (regression_head), a linear layer that maps the extracted features to two values, representing the coordinates of the pet nose. This architecture ensures that while we are using the powerful feature extraction capabilities of ResNet-18, we adapt the model to output continuous values pertinent to our keypoint detection task, rather than discrete class labels. The model is implemented in PyTorch, taking advantage of its dynamic computation graph and efficient tensor operations. The pre-trained weights of ResNet-18 (from ResNet18_Weights.DEFAULT) provide a solid starting point for training, imbuing our model with a prior understanding of image features.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2016.90

Image classification reference training scripts. GitHub. (n.d.). https://github.com/pytorch/vision/tree/main/references/classification#resnet

Resnet18¶. resnet18 - Torchvision main documentation. (n.d.). https://pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html

Step 3

Our training process utilized the following hyperparameters and hardware setup: Customized ResNet-18 model, adapted for keypoint detection. The model integrates a regression head in place of the original classification layer to output (x, y) coordinates. The model was trained for a total of 25 epochs. This number was chosen to ensure sufficient training for convergence without overfitting. We used a batch size of 64. This size created a balance between training speed and the effective utilization of GPU memory. The learning rate was set to 0.001, this rate was found to be optimal since it offered a good compromise between speed of convergence and stability of training. Mean Squared Error (MSE) loss was used to quantify the difference between the predicted and actual keypoints. Adam optimizer was employed for its efficiency in converging towards the global minimum.

Training was conducted on Google Colab, utilizing a Tesla V100 GPU. This setup provided the necessary computational power to process the dataset efficiently. The total training time was approximately 38 minutes and 28 seconds. The first epoch took significantly longer (approximately 22 minutes) due to initial setup and data loading times on Colab. Subsequent epochs averaged around 40 seconds each. Below in Figure 1 is the plot of training loss over the epochs. It illustrates a consistent decrease in loss, indicating effective learning and model improvement over time. The plot shows that the model's ability to predict keypoints improved significantly after the initial few epochs, with diminishing returns in loss reduction in later epochs.

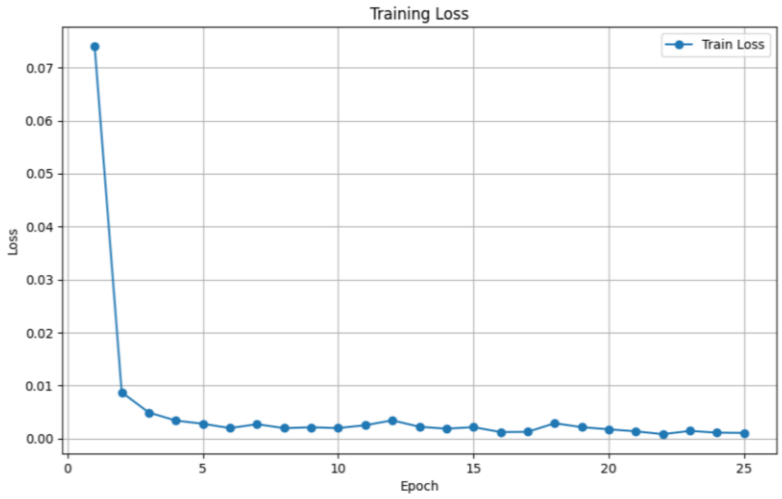

Figure 1: Generated Loss Plot

Step 4

To assess our model's performance, we employed both qualitative and quantitative methods. The testing script visualized the model's predictions by overlaying red dots on the test images at the predicted nose coordinates. For a deeper analysis, we calculated the Euclidean distance between the predicted and actual nose positions for each image, providing a robust quantitative measure of accuracy. Out of the test images, 10 were randomly selected, visualized with the keypoints, and saved for further qualitative analysis. These images can be seen in Figure 2 – Figure 11 below.

The test results revealed a minimum Euclidean distance of 0.0016 between the predicted and true coordinates, with a mean distance of 0.0599, a maximum of 0.3923, and a standard deviation of 0.0464. These statistics highlight the model's reliable performance in localizing pet noses, with most predictions closely aligning with the actual keypoints. The relatively low mean and standard deviation indicate consistent accuracy across the dataset, while the maximum distance gives insight into the upper limit of prediction deviation. In qualitative terms, the model effectively placed the keypoints near the nose regions in various scenarios, corroborating the quantitative findings and underscoring the model's utility in practical applications.

We conducted the tests on Google Colab using a Tesla V100 GPU. The model processed each image in an average time of 2.14 milliseconds, demonstrating its efficiency and suitability for real-time applications or processing large datasets. The entire test, including model loading, image processing, and result saving, was completed in just 15 seconds. While we encountered 'Clipping input data to the valid range for imshow' warnings, indicating potential issues with image normalization, this did not significantly impede the model's performance. Future investigations could explore this aspect further to potentially enhance accuracy.

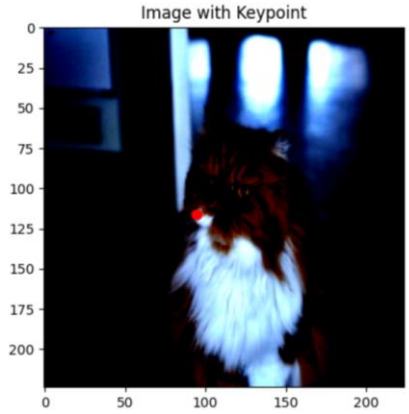

Figure 2: prediction:[0.38354513 0.38370174] for image predicted_keypoint_0.png

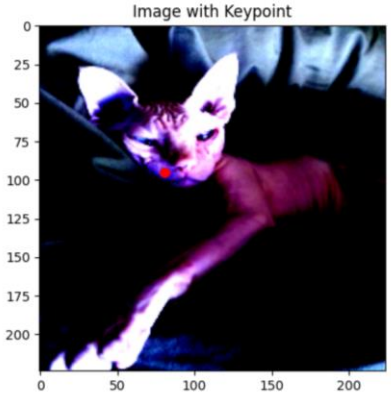

Figure 3: prediction:[0.48651552 0.43196344] for image predicted_keypoint_1.png

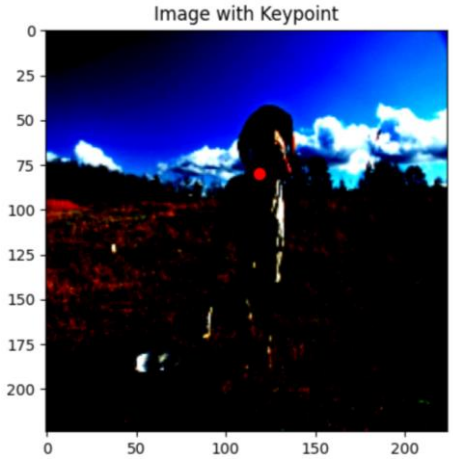

Figure 4: Prediction:[0.53223675 0.6725674] for image predicted_keypoint_2.png

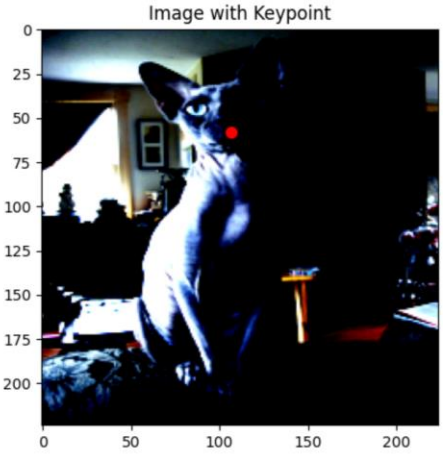

Figure 5: Prediction:[0.43873847 0.616233] for image predicted_keypoint_3.png

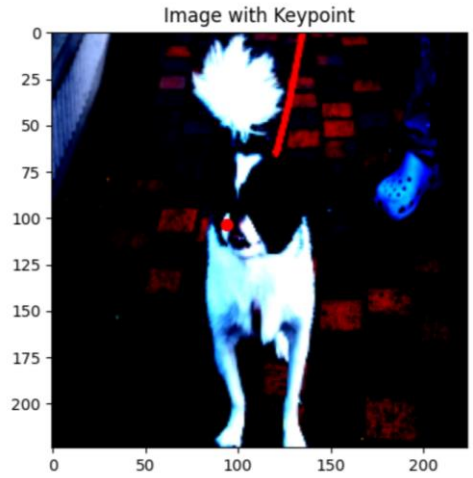

Figure 6: Prediction:[0.49065948 0.46243805] for image predicted_keypoint_4.png

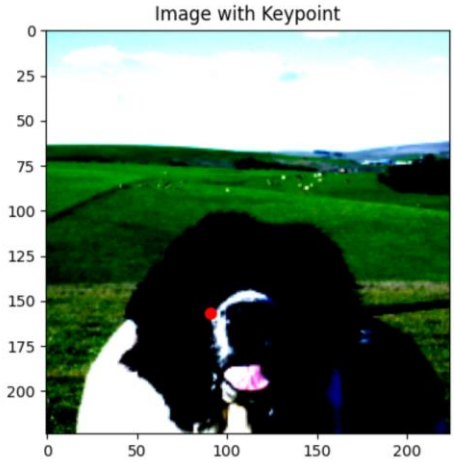

Figure 7: Prediction:[0.38949164 0.5152595] for image predicted_keypoint_5.png

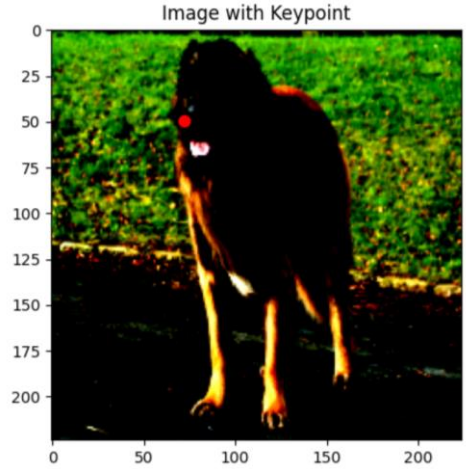

Figure 8: Prediction:[0.30372688 0.5705983] for image predicted_keypoint_6.png

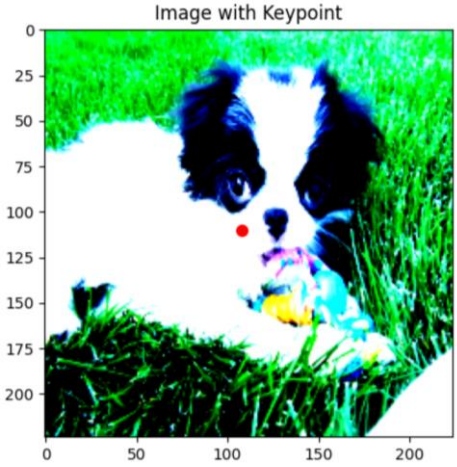

Figure 9: Prediction:[0.50866294 0.73291266] for image predicted_keypoint_7.png

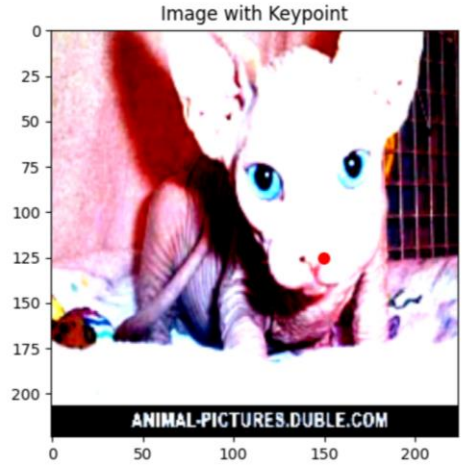

Figure 10: Prediction: [0.4352407 0.37272596] for image predicted_keypoint_8.png

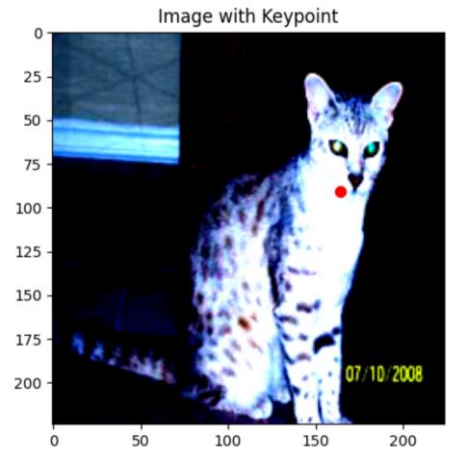

Figure 11: Prediction:[0.73246264 0.40416825] for image predicted_keypoint_9.png

Discussion

The system, based on a custom-tailored ResNet-18 architecture, performed adequately in pinpointing the noses of pets across a diverse dataset. The quantitative results, specifically the low mean distance of 0.0599 and a standard deviation of 0.0464, indicated a high degree of accuracy. Qualitatively, the red-dot visualizations on test images reinforced this, showing a close match between predicted and actual nose positions in most cases. Our expectations of achieving high accuracy in localization were mostly met. The system's rapid processing time (2.14 ms per image) exceeded our expectations, demonstrating its potential for real-time applications.

Choosing ResNet-18 as the base model was a strategic decision, balancing efficiency with complexity. However, customizing it for our specific task required significant experimentation, especially in fine-tuning hyperparameters like batch size and learning rate. The shift from a batch size of 32 to 64 and adjusting the learning rate were crucial in enhancing the model's learning efficiency. Initially, training on a CPU was time-consuming. Moving to Google Colab's Tesla V100 GPU dramatically improved training speed and efficiency. We encountered some issues related to image normalization and resizing. Adjusting the input size to 224x224 (optimal for ResNet) and applying appropriate normalization were critical steps. However, we noticed some images appeared darker post-processing, possibly due to the normalization or image handling methods used.

Ensuring accurate time measurement in the testing script posed a challenge. We refined the script to accurately calculate and report the time taken per image, which is crucial for understanding the model's practical applicability. Some challenges were faced in visualizing the results on Google Colab. Ensuring that the images with predicted keypoints were correctly displayed and saved required adjustments in the script.

We addressed model tuning by iteratively adjusting hyperparameters and assessing their impact on training performance. After switching to GPU training on Colab, the training process was much more efficient. Preprocessing steps were carefully revised to ensure image quality was maintained. The script was refined for accurate time measurement and effective visualization of results.

In summary, while the system met most of our expectations in terms of accuracy and efficiency, it highlighted areas for future improvement, particularly in image preprocessing and handling. The final result is a robust system that can determine the location of pet noses with moderate accuracy. The results demonstrate the effectiveness of our approach and the chosen architecture.