Make Test image similar to Training image

Submitted by – Nishant Jairath | Date – 23rd Nov 2024

Situation: For training a computer vision model on object detection (Leprosy in our use case) the training set images were collected in a controlled environment. On the other hand, the test images were obtained from patients in rural/backward areas with lack of proper lighting/zoom/focus and added noise/background. This difference in image quality led to degradation on model performance in the test conditions.

Task: Come up with approaches to make test image similar to training image by preprocessing them in order to save cost-time of retraining the model while improving the overall detection accuracy.

Assumptions: the trained model will remain the same and we only need to implement image processing techniques on top of it for improving the inferencing. The deployment architecture will also remain untouched and no changes will be made to it for optimization of time & accuracy. The end user is not going to change the device through the which the images are captured and therefore no suggestion towards that end is required.

Approach:

S.No	Methodology	Technology/Tool	References
1	**SAM (segment anything model) version 2** for instance segmentation and retrieval of the region of interest. This step is essentially doing the noise reduction by removing the background and enhancing the focus on particular region. Run the image through a series of preprocessing techniques to calibrate its quality with that of training set. Below are some of these – • Light adjustment – normalization of pixel intensities to standardize uneven distribution • Contrast correction – CLAHE (contrast limited adaptive histogram equalization) can be used for enhancing contrast • Color Jittering – tweak brightness, saturation and hue to match required RGB values • SRGAN/ESRGAN – enhanced super-resolution GAN for improving the image quality and zoom • Rescaling – rescale the segmented part of the image to make its size feasible for model inference	Python – opencv, SAM2, Mahotas, pytorch	Link 1 Link 2 Link 3 Link 4
2	**Domain Adaptation (DA)** – it is a type of transfer learning technique that allows a model trained on one domain to be used effectively in a different, but related, domain. It comprise of two categories - 1. Shallow DA methods: it further houses two ways i). Instance weighting - samples/instances in the source domain are assigned with different weights according to their relevance with target samples/instances	Python - Domain Adversarial Neural Network (DANN), CycleGAN,	Link 1 Link 2 Link 3 Link 4

	ii) Feature transformation. goal of feature transformation for DA is to construct a common/shared feature space for the source and target domains to reduce their distribution gap, based on various techniques such as low-rank representation 2. Deep DA methods: has multiple techniques - i). Transfer Component Analysis - utilizes a dimensionality reduction technique based on eigenvalue decomposition, where the projection matrix that best aligns the domains is calculated by minimizing a divergence measure between the source and target data distributions ii). Correlation Alignment (CORAL) - minimizes domain shift by aligning the second-order statistics of source and target distributions, without requiring any target labels Other state of the art techniques also exist such as Contrastive Semi-supervised learning for Cross Anatomy Domain Adaptation (CS-CADA).		
3	**Simple Linear Iterative Clustering (SLIC)** - creates superpixels by grouping pixels based on their color similarity and spatial proximity within the image. It will help in classifying the region of interest with better inferencing capability. To this we can add some of the methodologies described in S.No 1	Python – SLIC, mark_boundaries	Link 1 Link 2
4	**Neural Style Transfer** - allows us to generate an image with the same "content" as a base image, but with the "style" of our chosen picture. NST employs a pre-trained Convolutional Neural Network with added loss functions to transfer style from one image to another and synthesize a newly generated image with the features we want to add.	Python - ImageNet- VGG, TensorFlow,	Link 1 Link 2
5	**In-app Camera assist** - Define region of interest by guiding user to keep the infected area within a bounding box on camera screen and using these coordinates to cull it out of rest of the background for feeding into the model endpoint for inferencing. Follow the same image processing techniques as mentioned in S.No 1 for the final inferencing by model	Andriod/iOS, Python – opencv, Bokeh, Face touch	Link 1 Link 2 Link 3
6.	**RLHF (Reinforcement learning from human feedback)** - create a feedback loop to provide model with variations in dataset for and update the weights based on test environment. This will work as reinforcement learning algorithm where only the last layer of model weights are updated instead of retraining the whole model.	Python – OpenAI, Tensorflow	Link 1 Link 2 Link 3