Computer Vision 2023
(CSE344/ CSE544/ ECE344/ ECE544)
Assignment-1

Max Marks (UG/PG): 100 / 110 **Due Date**: 21/02/2023, 11:59 PM

Instructions

- Keep collaborations at high-level discussions. Copying/plagiarism will be dealt with strictly.
- Your submission should be a single zip file **Roll_Number_HW1.zip**. Include only the **relevant files** arranged with proper names. A single **.pdf report** explaining your codes with relevant graphs, visualization and solution to theory questions.
- Remember to **turn in** after uploading on Google Classroom. No justifications would be taken regarding this after the deadline.
- Start the assignment early. Resolve all your doubts from TAs during their office hours **two days before the deadline.**
- Kindly **document** your code. Don't forget to include all the necessary plots (loss, accuracy, confusion matrix, etc.) from **WandB** in your report.
- All [**PG**] questions, if any, are **optional for UG** students but are **mandatory for PG** students. UG students will get BONUS marks for solving that question.
- All [**BONUS**] questions, if any, are optional for all the students. As the name suggests, BONUS marks will be awarded to all the students who solve these questions.
- This is a **LONG** assignment. Please get started early to get a handle on the questions. There will be tutorials on ML/DL concepts that will be conducted online.

1. (40 points) **Image Classification**
   1. (5 points) Refer to the SVHN Dataset (Format 2).
      - (a) (1 point) Download train_32x32.mat. Use 20% of the training dataset for validation and 10% for testing. Initialize Weights & Biases (WandB)(Video Tutorial).
      - (b) (2 points) Create custom data loaders for all the splits (train, val and test) using PyTorch.
      - (c) (2 points) Visualize the data distribution across class labels for training and validation sets.
   2. (10 points) Training a CNN from scratch (Tutorial):

   (a) (3 points) Create a CNN architecture with 2 Convolution Layers having a kernel size of 3×3 and padding of 1. Use 32 feature maps for the first layer and 64 for the second. Finally add classification head to the conv layers. Use ReLU activation functions wherever applicable.
   (b) (4 points) Train the model using the Cross-Entropy Loss. Use wandb to log the training and validation losses and accuracies.
   (c) (3 points) Report the Accuracy and F1-Score on the test set. Also, log the confusion matrix using wandb.
   (d) **[PG]** (3.5 points) For each class in the test set, visualize any 3 images that were misclassified along with the predicted class label. Analyze why the model could possibly be failing in these cases. Is this due to the fact that image looks much similar to the predicted class than the actual ground truth or something else?

3. (10 points) Fine-tuning a pretrained model
   (a) (4 points) Train another classification model with a fine-tuned Resnet-18 (pre-trained on ImageNet) architecture using the same strategy used in Question 1.2.(b) and again use wandb for logging the loss and accuracy.
   (b) (3 points) Report the Accuracy and F1-Score on the test set. Also, log the confusion matrix using wandb.
   (c) (3 points) For deep neural networks, typically, the backbone is the part of a model (initial layers) that is used to extract *feature representations* (or simply features) from the raw input data, which can then used for classification or some other related task. These features are expressed as an n-dimensional vector, also known as a feature vector and the corresponding vector space is referred to as the feature space. As the training progresses and the classifier learns to classify the input, the data samples belonging to the same class lie closer to each other in the feature space than other data samples. For input samples from the training and validation sets, extract the feature vectors using the backbone (ResNet-18 in this case) and visualize them in the feature space using the tSNE plot in a 2-D Space. Also, visualize the tSNE plot of the validation set in a 3D-Space.

4. (10 points) Data augmentation techniques
   (a) (3 points) Use any 3 (or more) Data Augmentation techniques that are suitable for this problem. Remember that data augmentation techniques are used for synthetically adding more training data so that the model can train on more variety of data samples.
   (b) (4 points) Follow the same steps as in Question 1.3.(a) to train the model.
   (c) (3 points) Report the Accuracy and F1-Score on the test set. Also, log the confusion matrix using wandb.

5. **[BONUS]** (5 points) For each class in the test set, pick any 3 images that were misclassified using the basic CNN architecture (question 1.2). Report the euclidean distances in n-dimensional space (feature representation) of each misclassified sample from the centroid (mean) of the ground truth class as well as the predicted class.

Using the same approach, analyze the euclidean distances of the same samples using the other two trained models. If the samples are correctly classified using the other models, report the distance from the centroid of the predicted class as NA. Specify the names of the ground truth and predicted class.

6. (5 points) Compare and comment on the performance of all three models.

2. (30 points) **Image Segmentation**

   1. (5 points) Download the Segmentation Dataset. Refer to the VOC 2012 dataset to read about the label structure.
      - (a) (1 point) Download the training files. Use 20% of the training dataset for validation and 10% for testing. Initialize Weights & Biases (WandB).
      - (b) (2 points) Create dataloaders for all the splits (train, val and test) using PyTorch to load the images and their corresponding segmentation masks.
      - (c) (2 points) Visualize the data distribution across class labels for training and validation sets.
   2. (10 points) Fine-tune a segmentation model
      - (a) (4 points) Train fcn_resnet50 model using pre-defined network weights using an appropriate loss function. Use wandb for logging the loss and accuracy.
      - (b) (6 points) Report the classwise performance of the test set in terms of pixel-wise accuracy, F1-Score and IoU (Intersection Over Union). Also report precision, recall and average precision (AP). Use the IoUs within range [0, 1] with 0.1 interval size for computation of the above metrics. You may refer to this article to learn more about the evaluation of segmentation models. Include all your findings in the submitted report.
      - (c) **[PG]** (3.5 points) For each class in the test set, visualize any 3 images having IoU ≤ 0.5 along with the predicted and ground truth masks. Comment on why the model could possibly be failing in these cases with the help of IoU visualizations. Is the object occluded, is it being misclassified or is it due to the environment (surroundings) where the object is present?
   3. (10 points) Data augmentation techniques
      - (a) (2 points) Use any 2 (or more) Data Augmentation techniques that are suitable for this problem. Remember that data augmentation techniques are used for synthetically adding more training data so that the model can train on more variety of data samples.
      - (b) (4 points) Follow the same steps as in Question 2.2.(a) to train the model.
      - (c) (4 points) Report the performance on test set in terms of pixel-wise accuracy, F1-Score, IoU and AP.
   4. (5 points) Compare and comment on the performance of both the trained architectures.

3. (30 points) **Object Detection**

1. (5 points) Refer to the SVHN Dataset in yolo format. You can read about this dataset here (format 1).
   (a) (1 point) Download the dataset. Use 20% of the training dataset for validation and 10% for testing. Initialize Weights & Biases (WandB).
   (b) (2 points) Create data loaders for all the splits (train, val and test) using PyTorch to load the images and their corresponding bounding boxes.
   (c) (2 points) Visualize the data distribution across class labels for training and validation sets.
2. (10 points) Fine-tuning an object detector
   (a) (4 points) Train a YOLOv5 model. You can use the model from this repository. Use wandb for logging the loss and accuracy.
   Note: You are only allowed to use the model (pretrained) from this repo. You can not use dataloaders, training loops, metrics, etc... from it
   (b) (6 points) Report the performance of the test set in terms of classwise IoU, mean IoU (mIoU), Average Precision (AP) and mean Average Precision (mAP). Use the IoUs within range [0, 1] with 0.1 interval size for computation of the above metrics. Also plot Precision-Recall (PR) curve. Refer to this article to learn more about average precision in object detection.
   (c) **[PG]** (3 points) It is generally not a good idea to use all the IoU thresholds within the range [0, 1]. Instead, a good metric is AP@[0.50:0.5:0.95], i.e. AP calculated by using IoU thresholds within range [0.5, 0.95] with 0.5 as step interval size. Report the test performance using this metric. Also report the performance using AP50 and AP75. AP@α (also written as APα) is the value of AP with IoU $= \alpha$, i.e. AP50 (also written as AP@50 and AP@0.5) is the AP with IoU $= 0.5$.
3. (10 points) Data augmentation techniques
   (a) (3 points) Use any 3 (or more) Data Augmentation techniques that are suitable for this problem. Remember that data augmentation techniques are used for synthetically adding more training data so that the model can train on more variety of data samples.
   (b) (4 points) Follow the same steps as in Question 3.2.(a) to train the model.
   (c) (3 points) Report the performance of the test set in terms of IoU, AP and mAP, as done in question 3.2.(b).
4. (5 points) Compare and comment on the performance of both the trained architectures.