**PUPPYOPS - PS 1**

**Inference**

**About the data**

We have received a dataset consisting of 1966 images and annotations for semantic segmentation. This dataset is a subset of the cityscapes dataset.

We have 8 different cities and 19 classes which are road, sidewalk, building, person, car, tree etc.

The images are of high resolution 1024X2048 and in png format.

**Data Handling**

* Data Preparation - We have extracted train id images which are the ground truths of the images given. We have made a new data folder with images and train id images, in leftimg8bit and gtFine folders respectively.
* Data Preprocessing - We have then generated a new data folder “data”, with a train test validation split processing. The ratios are 0.6, 0.2, 0.2 respectively.

**Model Used**

*Encoder: DeepLabV3+*

DeepLabv3+ is a state-of-the-art convolutional neural network (CNN) designed primarily for semantic image segmentation. Its architecture consists of a powerful encoder-decoder structure combined with Atrous Spatial Pyramid Pooling (ASPP) modules.

*Decoder: EfficientNetV2M*

EfficientNetV2M excels in segmentation due to its efficient architecture and multi-scale feature representation. Its strong backbone provides state-of-the-art performance while enabling fast inference, making it ideal for real-time urban scene analysis like Cityscapes.

Reasons for choosing this model

* *Contextual Understanding*: DeepLabv3+ employs Atrous Spatial Pyramid Pooling (ASPP) to grasp contextual details across different scales, ensuring precise object boundaries.
* *Efficient Feature Extraction*: Its encoder-decoder architecture efficiently transforms image features into detailed segmentations, optimizing the use of captured information.
* *Proven Accuracy*: DeepLabv3+ consistently outperforms on challenging benchmarks like PASCAL VOC and Cityscapes, reflecting its reliability.
* *Ideal for Complex Tasks*

Diverse Scenes: Its strength lies in handling images with varied objects, scales, and intricate boundaries effectively.

Precision: For applications demanding pixel-level boundary accuracy, DeepLabv3+ delivers outstanding results.

* *Balanced Performance*

Computational Efficiency: While not as lightweight as MobileNet, DeepLabv3+ strikes a balance between model complexity and accuracy, making it a practical choice for our needs.

* *Considerations*

Real-Time Needs: For applications requiring swift segmentation, faster models like EfficientNet or MobileNet might be preferable.

Dataset Size: DeepLabv3+ thrives with ample data; with smaller datasets, its complexity may lead to overfitting, requiring careful training.

Model Configuration

* name: EfficientNetV2M-shuffle
* architecture: DeepLabV3plus
* backbone: EfficientNetV2
* unfreeze\_at: block6a\_expand\_activation
* input\_shape: [1024, 2048, 3]
* output\_stride: 32
* filters: [16, 32, 64, 128, 256]
* activation: leaky\_relu
* dropout\_rate: 0
* pretrained\_weights: No

**Training**

Training set - hamburg, darmstadt, bochum, aachen

Test set - erfurt, dusseldorf, cologne

Validation set - bremen

For training the following command has been used

python train.py --config ./config/Cityscapes.yaml

Accuracies: train\_mIoU - 0.81,val\_mIoU - 0.54, train\_loss - 0.58, val\_loss - 0.96

For evaluating the following command has been used

python evaluate.py --config ./config/Cityscapes.yaml

Accuracies(IoU): road - 0.86, building - 0.82, traffic sign - 0.52, vegetation - 0.87, sky - 0.92, car - 0.89 and so on

For predicting test the following command has been used

python predict.py --config ./config/Cityscapes.yaml --split "test"

For predicting val the following command has been used

python predict.py --config ./config/Cityscapes.yaml --split "val"

**Testing**

Test set - frankfurt, munster, zurich

We were given test data and then we put it in train file which is present in leftimg8bit folder

present in Images folder (Images -> leftimg8bit -> train -> 3 folders of test data given to us)

We have included a folder for testing custom images named “Run Test Images”, which has the

trained models saved as well as the script to run the test.

For testing use the following command

python testmax.py --config ./config/predict\_test.yaml --split “train”

Creating video results

By using Making\_Test\_Video\_results.ipynb we will convert the predicted images and segmented

images into a concatenated result file where original video is on the left and segmented video is

on the right side.

Here concatenation has been done so that it becomes easy for us to compare

Input has two paths:-

1st path: location of city in original dataset

2nd path: location of segmented images of that city (it will be stored in

\predictions\DeepLabV3plus\EfficientNetV2M-shuffle\train\rgb)

Destination path will be put according to your feasibility

**BONUS CHALLENGE - Real Time Segmentation using FastSam**

This code utilizes FastSAM, an instance segmentation model, to perform real-time segmentation on webcam video feeds. It uses the ultralytics library to load the FastSAM model and OpenCV for video capture and display. The model segments objects in the video frames, and the resulting annotations are overlaid on the original frames in real-time. Users can press 'q' to exit the application.The code harnesses the power of FastSAM, an advanced instance segmentation model, to deliver real-time object segmentation on webcam video feeds. At the core of this project is the ultralytics library, which seamlessly loads the FastSAM model. The code leverages OpenCV for efficient video capture and display capabilities.

As the webcam streams video frames, each frame is processed by the FastSAM model. The model performs intricate analysis to identify and segment objects within the frame. It leverages deep learning algorithms to distinguish objects from the background, accurately outlining their boundaries.

The resulting annotations, which represent the segmented objects, are then meticulously overlaid onto the original video frames. This overlay provides a visual representation of the objects detected in real-time. The code effectively merges the original frames with the annotations, creating an immersive and informative video output.

To ensure user control, the code incorporates a simple yet effective mechanism. By pressing the 'q' key, users can gracefully exit the application. This feature allows for seamless termination of the program, providing users with flexibility and control over their experience.

Note- The folder “nvvm” is copied from C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.8 to eradicate an Error related to JTL.