Project Documentation

- I've designed a complete new software architecture to make the project as easy as I can to be integrated or used. The project is very modular.
- I've used the idea of configuration files to avoid the hassle of modifying the code in case of hyperparameter changing.
- The project detects the number of classes during data preparation. Then, you'll change the number of classes in the config file to make it dynamic.

Example of a config file:

- *{*
- ***"train_data_file"**: **"/media/kokomind/New Volume/work/mostafa/acfr_apple_dataset/train.record"**,*
- ***"val_data_file"**: **"/media/kokomind/New Volume/work/mostafa/acfr_apple_dataset/val.record"**,*
- ***"image_size"**: [202, 308, 3], (Sizes of images in the dataset)*
- ***"shuffle"**: **true**, (To specify whether to shuffle the training set or not)*
- ***"buffer_size"**: 1024, (This is the maximum number of records to be loaded from disk into memory. It's useful for small memories, to avoid loading all the data in the memory)*
- ***"aug_delta_brightness"**: 0.2, (This is a data augmentation technique: changing the brightness of the input image randomly by a max factor of X)*
- ***"aug_flip_left_right"**: **true**, (This is a data augmentation technique: randomly flipping the input image horizontally)*
- ***"aug_delta_scale_pad_crop"**:[1.0, 1.0], (These are the lower and upper bounds for the scaling factor that is chosen randomly, then a zero padding is done and/or a crop)*
- ***"dataset_mean_values"**: [0, 0, 0], (These are the RGB mean values of the dataset used for normalization by subtraction. If not known, leave them to zeros)*

- ***"experiment_dir"**: **"./experiments/exp1_16"**, (Directory to save the model along with all the summaries and the checkpoints. This is important for making different experiments)*
- ***"max_num_checkpoints"**: 2, (Maximum number of checkpoints to save)*
- ***"pretrained_model_dir"**: **"./experiments/exp2_16_voc"**, (Directory from which a pretrained model exists. If there is no pretrained model, make sure that this is equal to "".)*

- ***"num_classes"**: 2, (Number of classes, which is very dynamic)*
- ***"num_epochs"**: 20, (Number of training epochs)*
- ***"batch_size"**: 8, (Batch size)*
- ***"output_stride"**: 16, (Currently supported strides 8 and 16 as proposed by the authors)*
- ***"width_multiplier"**: 1.0, (Width multiplier is used to control the number of filters. The lower the number, the faster the network at the cost of accuracy)*
- ***"weight_decay"**: 0.000000001, (This is the L2 regularization strength hyperparameter)*
- ***"dropout_keep_prob"**: 1.0, (If dropout is present, this is to control the probability of neuron keeping)*
- ***"enable_batchnorm"**: **true**, (This is to enable batch normalization, default is true)*
- ***"batchnorm_decay"**: 0.9997, (This is the decay parameter used for running mean, and average of batch normalization)*
- ***"initial_learning_rate"**: 5e-5, (The learning rate to start training from)*
- ***"final_learning_rate"**:1e-5, (The learning rate to end the training with)*
- ***"learning_rate_decay_power"**:0.9, (The decay power for the learning rate)*

- ***"data_format"**: **"NCHW"**, ('NCHW' or 'NHWC'. It's proved that 'NCHW' provides a performance boost for GPUs. However, 'NHWC' is the only one that can be used for CPU computations)*
- ***"max_num_tensorboard_images"**: 50, (Maximum number of images to log in tensorboard)*
- ***"log_every"**: 10, (Display logging in the command line every X number of iterations)*
- ***"tensorboard_update_every"**: 100, (Update tensorboard logging every X number of iterations)*

- ***"test_video_file"**: **"/media/kokomind/New Volume/work/mostafa/test_video.mp4"**, (The test video filename)*
- ***"save_output_video"**: **true**, (This is an option to save the output video or not. Not saving it leads to a higher FPS)*
- ***"dump_frames_every"**: 1, (Dumping frames to disk every X frames. Higher number leads to saving multiple frames at once which may increase FPS on machines that have an SSD)*
- ***"output_video_file"**: **"/media/kokomind/New Volume/work/mostafa/test_video_output.mp4"**, (The output video filename)*
- ***"test_model_timestamp_directory"**: **"1539534611"** (The exported model subdirectory which is created automatically by tensorflow under the provided experiment directory. To get it, look at the exported model directory inside the experiment directory)*

- The code is commented line by line, method by method, class by class to make it very documented and scalable.
- The project supports NCHW and NHWC data format. As stated by TensorFlow and its competitors, NCHW provides a performance boost during training.
- The project is built based on TFRecords due to its performance advantage. This is to avoid loading all the data into memory. It's very useful when large datasets are used.
- The project is built based on TFEstimator to support training on a distributed system.
- The project supports an output stride of 8 and 16 as proposed by the authors. [stride 8] is very computationally expensive but provides a higher accuracy than [stride 16]
- The project supports TensorBoard for visualization. It logs [loss, pixel-wise accuracy, mean-iou, learning rate, and number of steps / sec] for training and evaluation. It also uses TFLogger to log everything to the command line. It's very easy to be logged into a file instead.
- The project supports a decaying learning rate as proposed by the authors of DeepLab-V3+ to avoid overfitting.
- The project supports data augmentation to overcome overfitting. Data augmentation methods are: Random Brightness, Random Flipping, Random Resizing and Cropping or Padding.
- The project supports loading from a previously pretrained model to perform transfer learning.

- There are four important files which run in order to get a ready model for testing:
  1. create_tf_record_images.py : to construct and prepare the dataset. The dataset should be of the following structure:
     - Total input RGB images in [name.png] format.
     - Total input segmentation images in [name_L.png] format.
     - A train.txt file specifying the training image names, one per line
       Name1
       Name2

-
-
- A val.txt file specifying the validation image names, one per line

Then, the python file uses a config file of the following structure:

```
{
- # Enable only if validation data exists
-  "VALIDATION_EXISTS" : true,
- # Path to the directory which will store the TFRecord train file
-  "TRAIN_TF_RECORD_NAME" : "/media/kokomind/train.record",
- # Path to the directory which will store the TFRecord validation file
- "VAL_TF_RECORD_NAME" : "/media/kokomind/val.record",
- # Path to the file containing the training data
- "TRAIN_DATA_LIST_NAME" : "/media/kokomind/New Volume/work/mostafa/acfr_apple_dataset/train.txt",
- # Path to the file containing the validation data
- "VAL_DATA_LIST_NAME" : "/media/kokomind/New Volume/work/mostafa/acfr_apple_dataset/val.txt",
  # Path to the directory containing the training images
- "IMAGE_DATA_DIR" : "/media/kokomind/New Volume/work/mostafa/acfr_apple_dataset/images",
  # Path to the directory containing the training labels
- "LABEL_DATA_DIR" : "/media/kokomind/New Volume/work/mostafa/acfr_apple_dataset/segmentations",
  # Resize Image Height and Width
- "OUTPUT_HEIGHT" : 202,
- "OUTPUT_WIDTH" : 308,
  # Color coded or label coded. If color coded, then an RGB image with colors will be read. Else, a label coded with
  labels [0,1,2, etc.] will be read.
- "LABEL_COLOR_CODED" : true,
  # This is the boundaries label that should be ignored. If equals -1, no labels are ignored.
- "IGNORE_LABEL" : 255
- }
```

Usage: python create_tf_record_images.py --config [config_filename]
Example: python create_tf_record_images.py --config config/voc_vanilla_dataset.json

2. train.py : to train the model and save it to a provided experiment directory.
   Usage: python train.py --config [config_filename]
   Example: python train.py --config config/train_exp1.json
3. export_inference_graph.py : to export the model into a deployment version.
   Usage: python export_inference_graph.py --config [config_filename]
   Example: python export_inference_graph.py --config config/train_exp1.json
4. inference_api.py: This API is used to make inference easier than ever. (inference_video.py and inference_webcam.py are sample applications that use this API, run any of them using the same procedure ***python inference_video.py --config [config_filename]***). You just load the model using ExportedModel(model_path) instance. Then, you can call model.predict() and pass the input. You'll receive the corresponding [input after resize, predicted classes, predicted classes after decoding into a visible image]

- Output: The network runs on average with 45FPS on 1070TI at Stride 16 and resolution [202, 308]. Note that there are some pre-processing and post-processing to get decoded colored images from the predicted classes.

Annotation Requirements

- The standard of label annotation in segmentation is based on PASCAL VOC colors (256 normal color palette) and is of the following format:
  [(0, 0, 0), (128, 0, 0), (0, 128, 0), (128, 128, 0), (0, 0, 128), (128, 0, 128),
  (0, 128, 128), (128, 128, 128), (64, 0, 0), (192, 0, 0), (64, 128, 0),
  (192, 128, 0), (64, 0, 128), (192, 0, 128), (64, 128, 128), (192, 128, 128),
  (0, 64, 0), (128, 64, 0), (0, 192, 0), (128, 192, 0), (0, 64, 128)]
  Where the first element is RGB corresponding to class 0 (black), the second is the one corresponding to class 1 (red), and so on.
- As a reference, dataset should be of the structure mentioned above and found in the *dataset.zip* example for the apple dataset.

Tensorboard Visualization

For visualization purposes, run tensorboard from the command line and open it in the experiment directory. Tensorboard views the loss, mean iou, accuracy, etc. curves. It also views the training images output and validation images output.

tensorboard --logdir=[experiment directory here] --port=[your port for example 6006]

Then, go from the browser to the address *localhost:portname.*

Essential Dependencies

- Python 3.6
- Tensorflow 1.11
- Scikit-video (using pip install)
- ffmpeg
  On Windows, to work with ffmpeg, make sure that you follow the following link for installation.
  http://blog.gregzaal.com/how-to-install-ffmpeg-on-windows/
  Then, install scikit-video package.

Hyperparameters that achieved Highest mIOU on Apple Val. Dataset

```
"batch_size": 8,
"weight_decay": 0.000000001,
"initial_learning_rate": 5e-5,
"final_learning_rate":1e-5
```