Weekly Work Summary

Group 2 - Conditional GAN

Week 1,2 -07.05.2019:

- Read the CycleGAN paper
- Looked into the official implementation and tested it
- Developed ideas how to realize the conditioning of the GAN:
    - Adding a new one-hot-encoded channel to the input image
    - Adding the one-hot-encoded information into the model (e.g. between Encoder and Decoder)

Week 3 -14.05.2019:

- Changing the existing code to implement the binary “add conditional channel” approach:
    - Added function to add conditional channels. Binary case is also one hot encoded (adding two extra layers) to make future scaling easier.
    - Add argument to parser to manually define number of added channels and index of channel to be filled with 1s.
    - Change CycleGAN architecture, so G_A and G_B are becoming the same generator, e.g. changing G_B to G_A. (G_A=G and G_B=F in paper)
    - Change the Generator to take an arbitrary number of channels as input, before it could only take RGB or greyscale images.
- Problem: Conditioning doesn’t work.
    - “Input A conditioned to B” still outputs fake B and “Input B conditioned to A” still outputs fake A
    - “Input A conditioned to A” outputs fake B instead of A (also for B-case)
    - In other words, the network just creates fake images from the other domain. It never creates fake images of the same domain.
    - Hypothesis: The network learns to ignore the conditional channels due to no training images “Input A conditioned to A”. It learns that the conditional channels don’t bring any information

Week 4 -21.05.2019:

- CycleGAN: Had to make major changes in model cycle_gan_model.py to make it able to condition on the extra channels provided. However, there seems to be some logical error as results do not reflect conditioning yet. Looking into it.
- MUNIT:
    - Getting familiar by reading paper
    - Use an implemented code, Problems:
        - The original implementation is based on Pytorch 0.4.1, failed to create a conda environment with Pytorch 0.4.1
            - Error message: ‘Disk quota exceeded’
        - Unofficial Tensorflow implementation, Problem: Tensorflow GPU-Version doesn’t work on TUM server
            - Solution 1: Train on CPU, very slow by factor 100
            - Solution 2: Use Google Colab:

        - https://colab.research.google.com/drive/1jWHJ9B-kZzbKokfedVUDNhqXokFioa8h
        - https://github.com/MichinariNukazawa/MUNIT-Tensorflow
    - Training the model on a very small dataset (10 pics) to overfit:
        - TrainA: 10 pics from w2
        - Train B: 5 pics from w4 and 5 pics from w6
        - After training, use a guided example from w4 and w6 respectively to create w2->w4/w6

Week 5 -28.05.2019:

- MUNIT: Trained on extended/full dataset for longer (100.000 iterations vs. 40.000 from before)

Week 6 -04.06.2019:

- StarGAN: Trained on Simulated dataset
    - Worked very well
- Oxford Robotcar Dataset: Preprocessing
    - Script for scraping the datasets automatically
        - 1000 images for every of the four single tags: night, rainy, snowy, sunny
    - Converting Bayes Image into RGB
    - Crop into 256x256 image
- StarGAN: Trained on Oxford Robotcar Dataset
    - Worked quite well
    - Still needs a lot more training (until now 100.000 iterations)
    - Colors are mostly correct, details like snow or water on the streets are still in the learning process
- General: Cleaned our Gitlab Repo
    - Master branch now only contains presentations and summary
    - StarGAN is in ‘stargan’ branch
        - Google Colab link can be found in the README of stargan
    - Uploaded MUNIT

Week 7 -11.06.2019:

- Working on continuous StarGAN:
    - All work is saved in branch “c_stargan”
- Sanity Check if StarGAN can do projective transformation, results:
    - First check failed due to random horizontal flip as augmentation
    - Second check successful
- Changed the code to make StarGAN continuous, in this first version, the **target variable is absolute not relative**, so it should be easier to train, as the model doesn’t need to figure out the current position/can make a position invariant latent content representation.
    - First training was quite successful, trained for 100.000 iterations
        - results can be seen in the Gitlab
        - test also with conditional values, that are unseen during training
        - Wrong cropping (cropped in, instead of just changing resolution)
    - Second test still on the run with correct cropping

Week 8 -18.06.2019:

- Changed test method, to make it easier to test different values
- Change of StarGans to make it work on aligned data
- Intruduction of reconstruction loss for target image (which was not possible earleir)
- Conditioning on relative distance of cameras instead of absolute classes
- Tested various combination of above mentioned changes to debug results
- Code available at https://gitlab.vision.in.tum.de/s0045/stargan
- More evaluations of the model
    - Evaluation on correct crop size
    - Evaluation with unseen values
    - Evaluation with smaller step size

Week 9, 10 – 09.07.2019

- Conducted following experiments.
    1. relative distance
    2. relative distance with no cycle loss
    3. relative distance with no cycle loss and no classification loss
    4. relative distance with no cycle loss and classification loss for both directions
    5. relative distance with no cycle loss and classification loss for both directions and a missing camera (To cope with testing on unseen cameras)
    6. relative distance with no cycle loss and source (real/fake) loss
- Implementation of L1/L2 metric for quantitative evaluation
- Made the code from StarGAN importable into other python applications
- Made a flask server application to demonstrate model effects on a real image with a slider for distance. (This app is available on http://efe5425e.ngrok.io/ for next 24 hours and uses model number 3 mentioned above which is our most successful model yet trained on 150k iterations)

Week 11 -16.07.2019

- Following experiments:
    1. No Discriminator
    2. No L1 Loss
    3. No Source Loss, only Classification Loss and
        - 1 * L1 Loss
        - 100 * L1 Loss
    4. Training on full dataset with different weather conditions

Week 12 -23.07.2019

- Training the model on new dataset with more randomness (the car driving a serpentine line).
- Testing on more shifting values.