# Cycle-GAN

•••

github.com/gvsakashb/cyc-gan

Akash

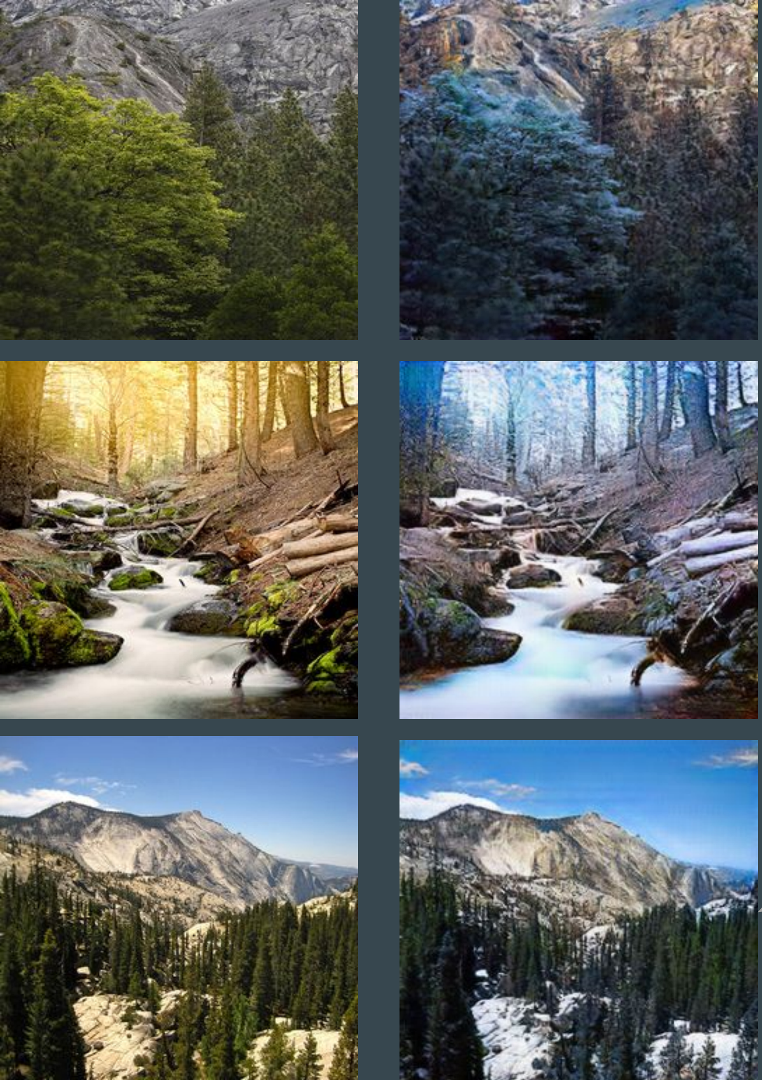
# Contents

- Re-implementation
- Changes and tweaks
- Metrics and output
- Status update
- Deliverables

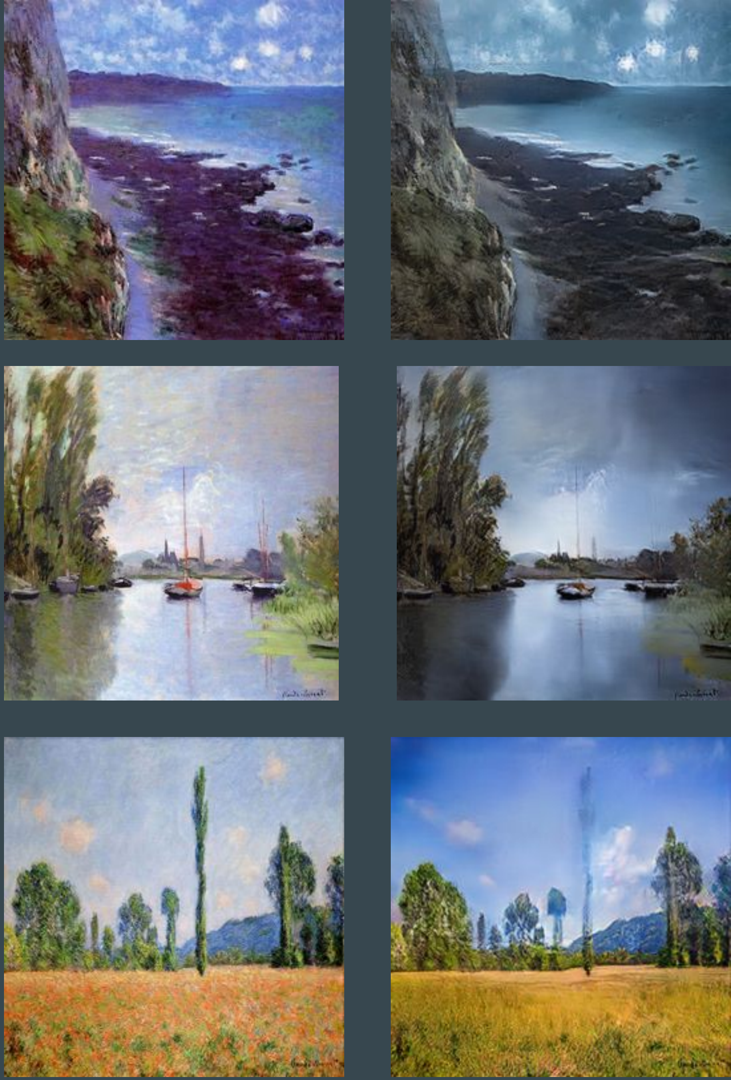
# Re-implementation & extensibility

1. Achieved images comparable to best results and reconstructions.
    a. Improvements in *horse2zebra* and *yosemite* images generated after changes in util, options directory for helper files.
        i. Reconstruction errors and results have also been generated for the report.
2. Comparison of various models (parameter tuning) and the images,
    a. Best metrics for content_loss, --netG, normalization, etc. have been illustrated.
3. Tabulated the losses for D, G and developed new models with resized convolutions.
    a. Initial changes to train_options.py file and data directory's codebase.
    b. A separate implementation from scratch, mainly to assess and study the architecture..

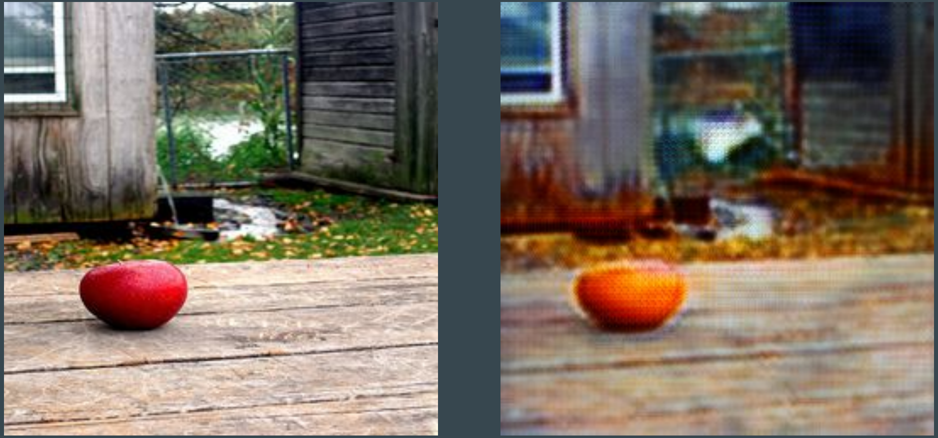
# Best results

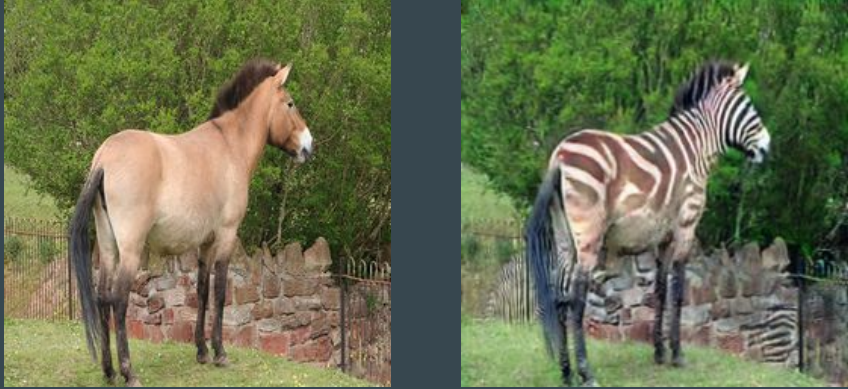


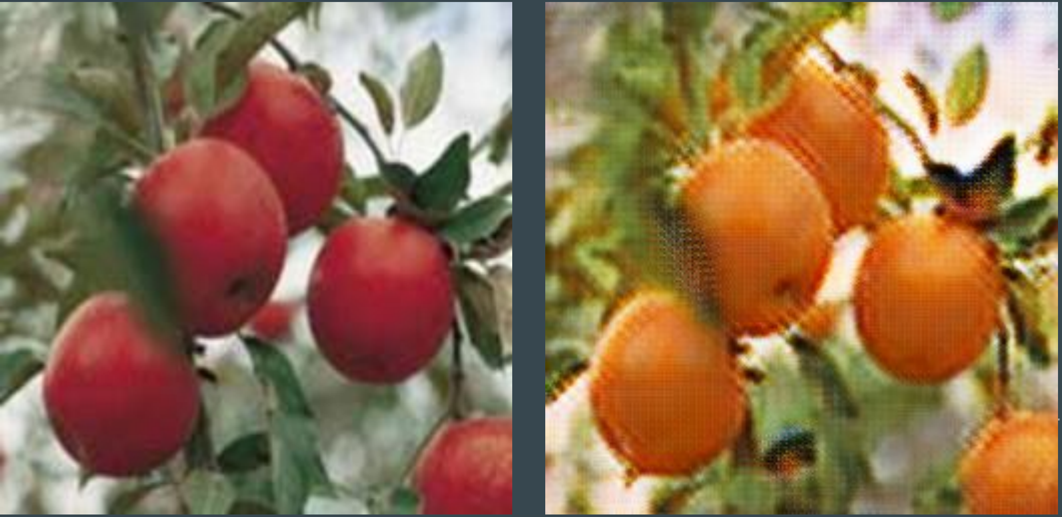
Summer2Winter_Yosemite

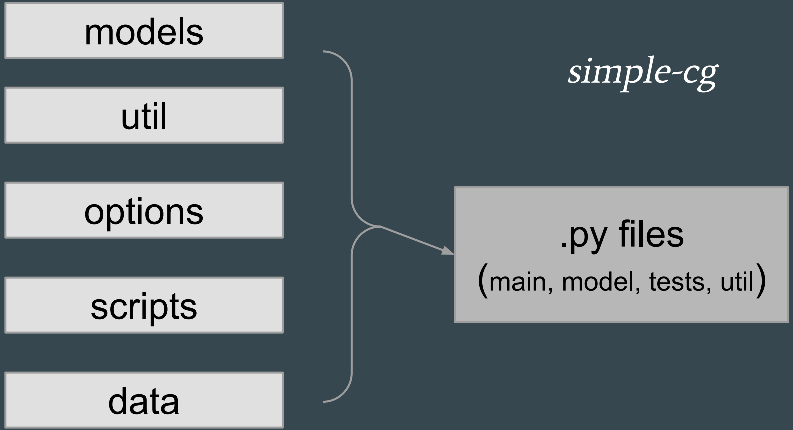


Monet2Photo









models

util

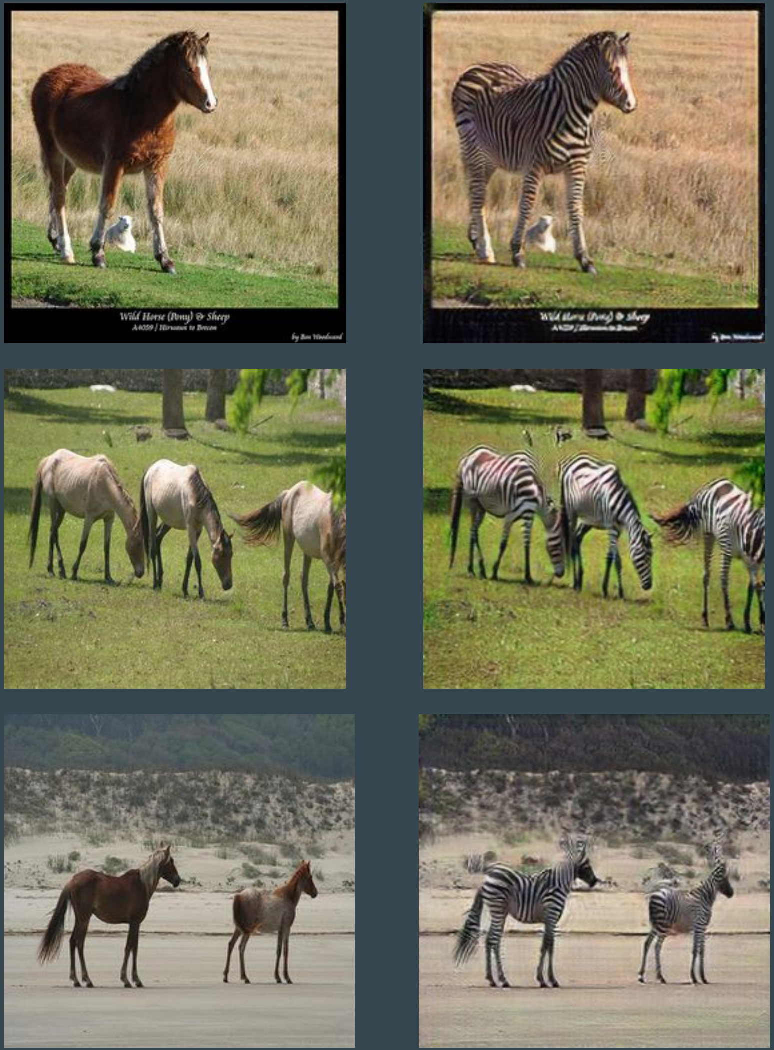
options

scripts

data

.py files
(main, model, tests, util)

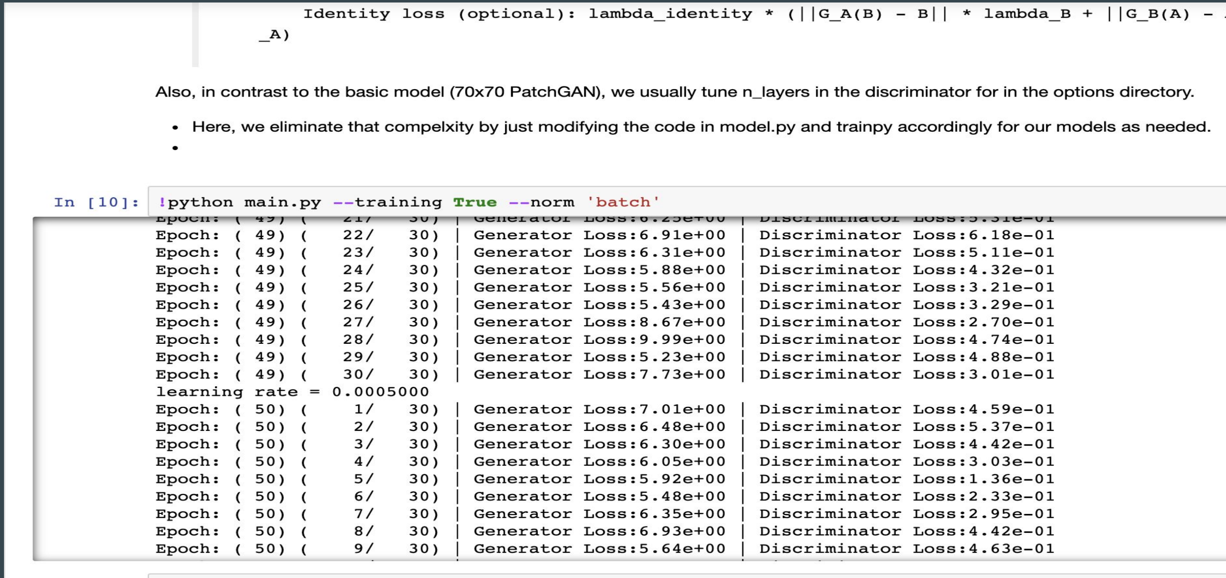
*simple-cg*



# Tweaks

- The original repo has been explored by modifying various parameters in each iteration of the datasets.
- A new directory (simple-cg) has been developed with trained models and submitted.
- Improvements observed include clarity, dehazing and reducing noise.
- Developed a separate model to log and save the adversarial losses and the analysis for various runs changing the  lr-policy & --netG parameters.

# Metrics



```
Identity loss (optional): lambda_identity * (||G_A(B) - B|| * lambda_B + ||G_B(A) -
_A)
```

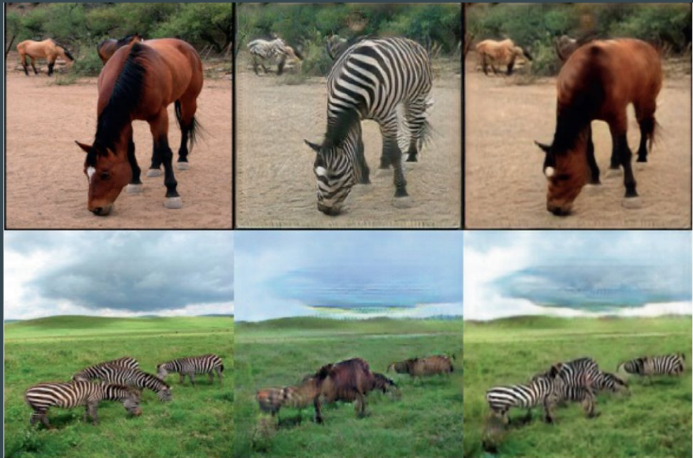
Also, in contrast to the basic model (70x70 PatchGAN), we usually tune n_layers in the discriminator for in the options directory.

- Here, we eliminate that compelxity by just modifying the code in model.py and trainpy accordingly for our models as needed.
-

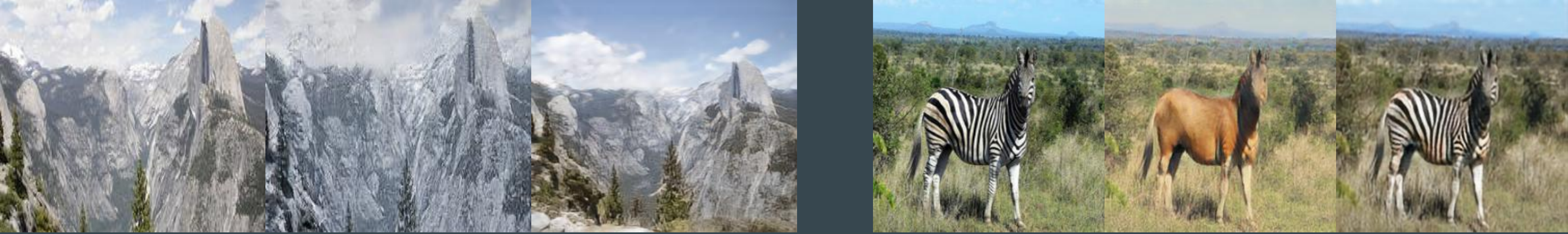
```
In [10]: !python main.py --training True --norm 'batch'
```

```
Epoch: ( 49) (   21/   30) | Generator Loss:6.25e+00 | Discriminator Loss:5.31e-01
Epoch: ( 49) (   22/   30) | Generator Loss:6.91e+00 | Discriminator Loss:6.18e-01
Epoch: ( 49) (   23/   30) | Generator Loss:6.31e+00 | Discriminator Loss:5.11e-01
Epoch: ( 49) (   24/   30) | Generator Loss:5.88e+00 | Discriminator Loss:4.32e-01
Epoch: ( 49) (   25/   30) | Generator Loss:5.56e+00 | Discriminator Loss:3.21e-01
Epoch: ( 49) (   26/   30) | Generator Loss:5.43e+00 | Discriminator Loss:3.29e-01
Epoch: ( 49) (   27/   30) | Generator Loss:8.67e+00 | Discriminator Loss:2.70e-01
Epoch: ( 49) (   28/   30) | Generator Loss:9.99e+00 | Discriminator Loss:4.74e-01
Epoch: ( 49) (   29/   30) | Generator Loss:5.23e+00 | Discriminator Loss:4.88e-01
Epoch: ( 49) (   30/   30) | Generator Loss:7.73e+00 | Discriminator Loss:3.01e-01
learning rate = 0.0005000
Epoch: ( 50) (    1/   30) | Generator Loss:7.01e+00 | Discriminator Loss:4.59e-01
Epoch: ( 50) (    2/   30) | Generator Loss:6.48e+00 | Discriminator Loss:5.37e-01
Epoch: ( 50) (    3/   30) | Generator Loss:6.30e+00 | Discriminator Loss:4.42e-01
Epoch: ( 50) (    4/   30) | Generator Loss:6.05e+00 | Discriminator Loss:3.03e-01
Epoch: ( 50) (    5/   30) | Generator Loss:5.92e+00 | Discriminator Loss:1.36e-01
Epoch: ( 50) (    6/   30) | Generator Loss:5.48e+00 | Discriminator Loss:2.33e-01
Epoch: ( 50) (    7/   30) | Generator Loss:6.35e+00 | Discriminator Loss:2.95e-01
Epoch: ( 50) (    8/   30) | Generator Loss:6.93e+00 | Discriminator Loss:4.42e-01
Epoch: ( 50) (    9/   30) | Generator Loss:5.64e+00 | Discriminator Loss:4.63e-01
```

| **Task** | **dataset** | **metric** | **value** |
|---|---|---|---|
| Img-to-img translation | *ukiyoe* | Class IoU score | 0.1012 |
| Img-to-img translation | *monet-photo* | Quality | 0.408 |
| Img-to-img translation | *horse-zebra* | Per-pixel accuracy /ablation study score | 0.52 |
| Img-to-img translation | *horse-zebra* | Per-class accuracy | 0.17 |

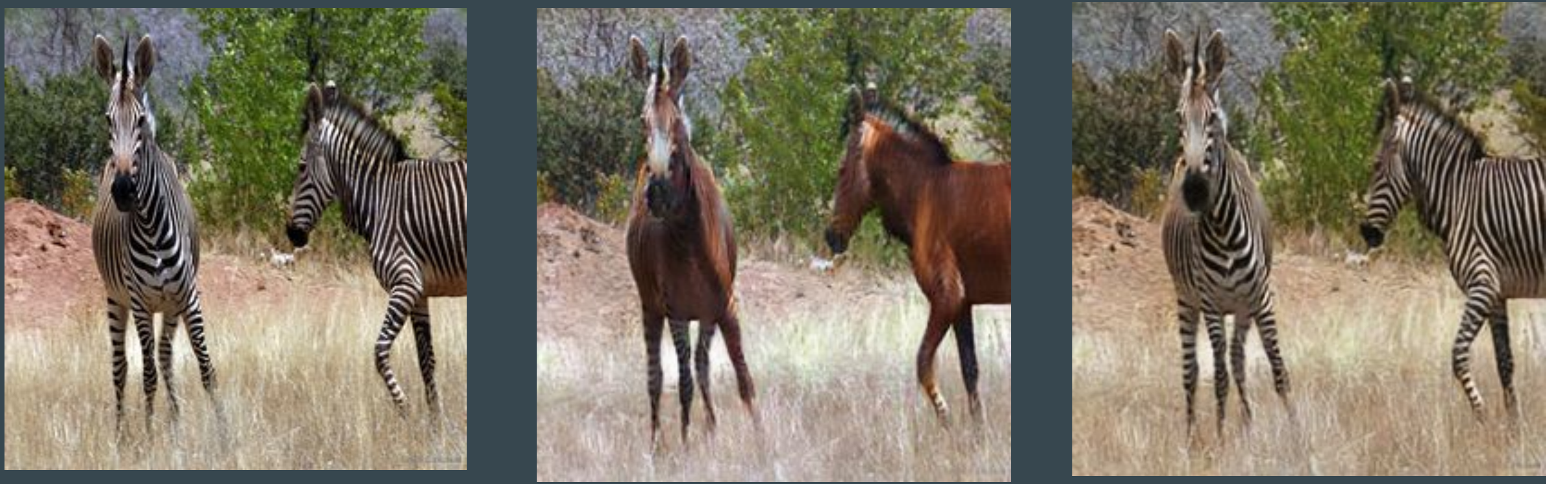


# Reconstruction (results)



Re-implementation results ^

From newer implementation:



# Status :

1. Successful implementations and models for h2z, monet, ukiyoe and seasonal datasets have been done.
2. A simpler version of cycle-gan model, with easier access to parameter tuning has been developed and trained.
   a. Two approaches with better latency, to compare batch vs instance norm.
3. Comparative analysis of loss of G & D against each epoch in training has been tabulated.
4. Improved latency with changes to train & model.py files.

# Issues:

- Setup and latency of the nvidia-docker image for the official docker file, resorted to AWS and then deploying to Docker.
- Object transfiguration and some errors in hue and tint of images (especially Ukiyo-e)

# Deliverables:

- Repository with both versions
  - Re-implementation & simple-cg directories
  - Trained notebooks and code
- Deep Learning AMI
  - Ready to deploy / trained models for each dataset (h2z, monet, seasonal & art styles)
  - Simple-cg implementation and tabulated metrics(losses)
  - Access key for AMI and readme for opening the EC2 instance
- Final report & documentation, results

- *Additional: Working on scaling AMI upon Docker image* (with accompanying file)*