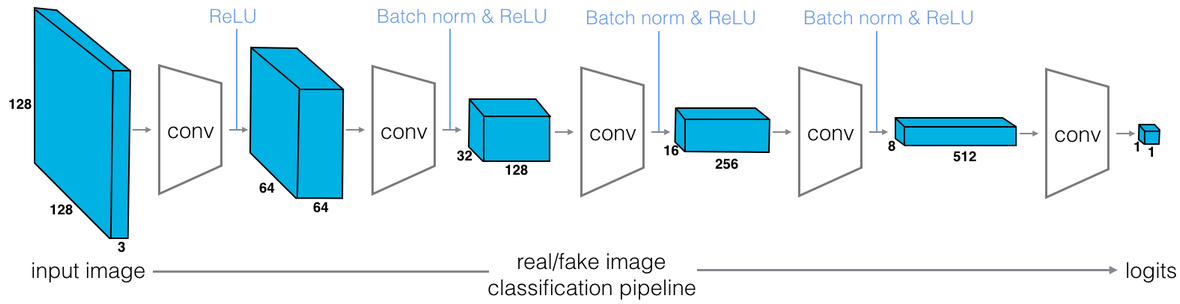
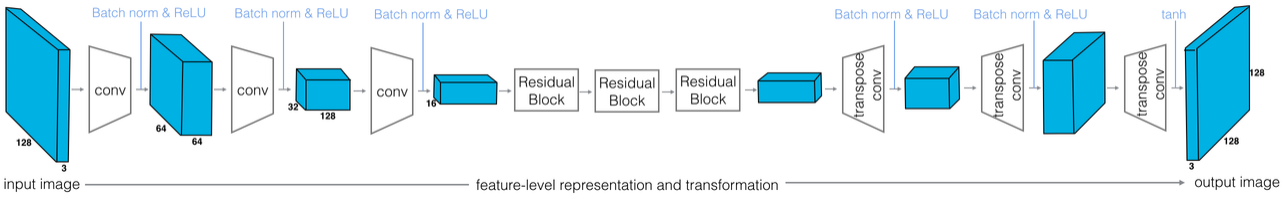
Cycle GAN implementation using Pytorch

- Load image dataset using DataLoader
- **Discriminator**



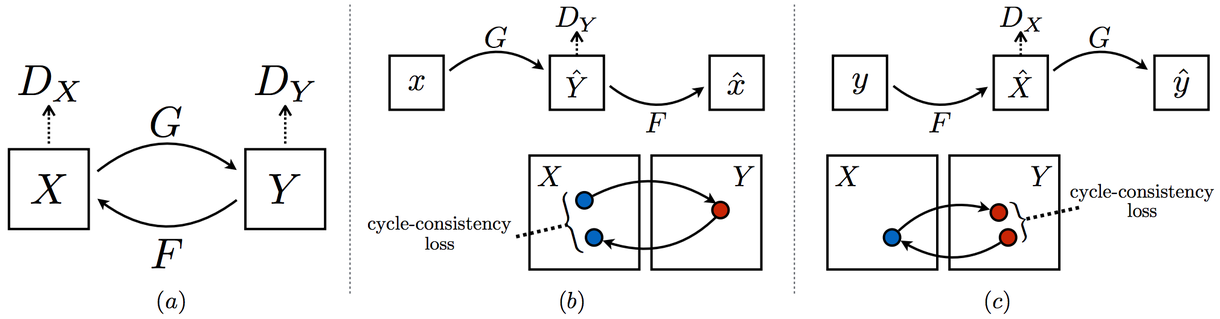
Using kernel size = 4 and stride = 2, every layer the size of the image is half, except for last layer.
128x128x3 => 64x64x64 => 32x32x128 => 16x16x254 => 8x8x512 => 1x1x1 (?).

- **Generator**

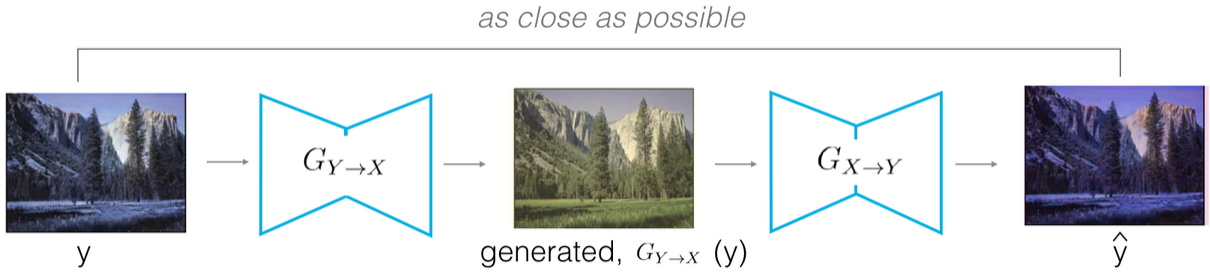


Also using layers like discriminator, the generator constructs layers until 16x16x256 and use several residual blocks and do transpose of convolutional layers to generate an output image.

- Complete **Architecture**
  - 2 discriminators: DX and DY
  - 2 generators: GX→Y and GY→X
- **Losses**



- 2 Discriminator losses: mean squared error
  - sum of real images loss and fake images (generated by G) loss
  - DX and DY loss computed separately
- Generator loss: **sum** of generator losses + forward and backward cycle-consistency loss
  - *Note: G loss on fake image is D loss on real image (because G tries to generate images that look as real as possible)
  - image => GY→X => GX→Y => compare reconstructed with original image



- Some **Tips** and **Improvements**
  - 2 Discriminators should have same loss levels
  - Generator loss should be higher and decrease a lot at the start. If it fluctuates, then decrease learning rate / change weight of cycle consistency loss
  - Use 256x256 for high-res data
  - Add color-based loss term
  - Pix2Pix Architecture
  - More suitable for stylistic transformation (not geometric)