ADLxMLDS Final Report - Image Inpainting

Motivation

Problem definition

- Image inpainting is the process of reconstructing lost or deteriorated parts of images.
- We remove the lower half of the images, and aim to recover the lost part or generate reasonable images.

Dataset

- Link: http://image-net.org/small/download.php (http://image-net.org/small/download.php)
- We use the 32x32 images from image-net
  - 1281149 images for training
  - 50000 images for testing

Models

Pix2Pix

(pic-model)

- Pix2Pix is a conditional gan which conditioned on an input image and generate a corresponding output image
- In our case, the input image is an image whose lower-half part is sliced. We want to generate the full image
- Generator: U-Net based
  - U-Net is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.
  - Since there are a great deal of low-level information shared between the input and output, if there are skip connections, these information can be shuttled directly across the net.

(pic-Unet)

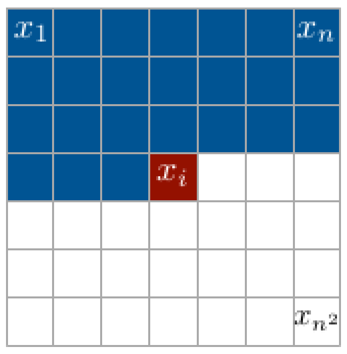
- Discriminator: convolutional PatchGAN classifier
  - penalize structure at the scale of image patches: classify if each N × N patch in an image is real or fake
  - Since N can be smaller than image size, PatchGAN has fewer parameters, can run faster and be applied on arbitrarily large images.
- Loss function
  (pic-loss_GAN)
  (pic-loss_L1)
  (pic-loss_final)
- L1 captures the low frequencies, while GAN captures high frequencies
- Does not have a Gaussian Noise z. Pix2pix provide noise only in the form of dropout

PixelCNN++

Original PixelCNN: a generative model of images with tractable likelihoods.

- The probabilty density function of an image x is factorized into all its pixels as $p(x) = \Pi_i p(x_i | x_{<i})$

- $x_{<i}$ consists of pixels to the up and the the left of x_i, which is called the “context” of x_i



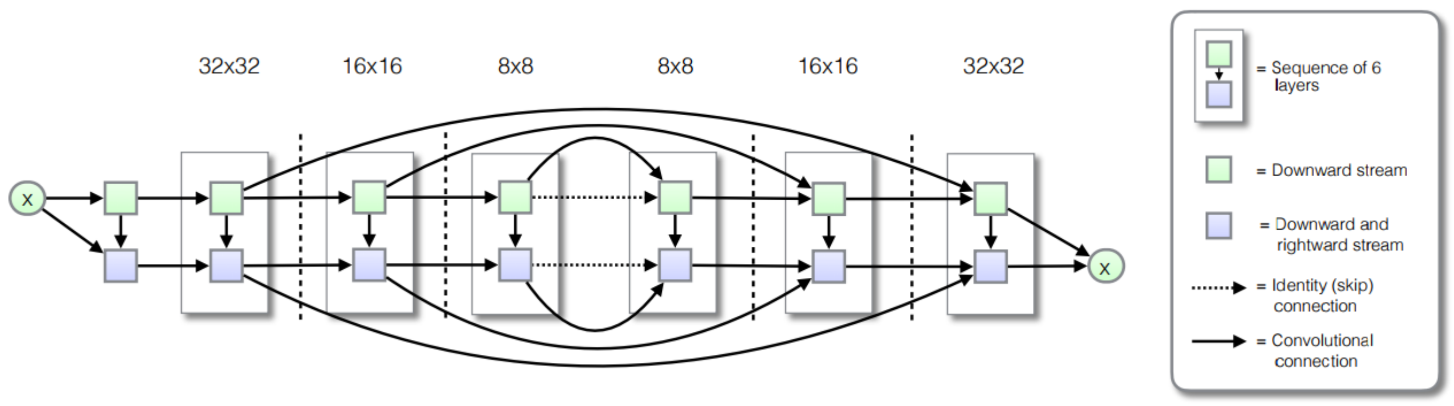
Context

- Thus, the generation proceeds row by row and pixel by pixel.
- When considering RGB channels, the distribution is factorized as $p(x_i|x_{<i}) = p(x_{i,R}|x_{<i})p(x_{i,G}|x_{<i}, x_{i,R})p(x_{i,B}|x_{<i}, x_{i,R}, x_{i,G})$

- Training the model maximizes the probability on training images. The ditribution over each pixel can be computed parallelly during training, as opposed to the sequential process during generation.
- Pixels as discrete variables: each channel variable $x_{i,*}$ takes one of 256 integer values and is modeled with a softmax layer.
- Using residual connections helps training this deep model with up to twelve layers.

PixelCNN++: simplify the structure and improve its performance.

- Use discretized logistic mixture likelihood:
    - The 256-way softmax is poor in memory usage, slow in training, and difficult to generalize to unseen pixel values.
    - Use a simple continuous distribution to model the sub-pixel color intensity ν, and then round it to integer x, the nearest 8-bit representation.
    - Consider mixture of logistic distribution: $\nu \sim \sum_{k=1}^{K} \pi_k \text{logistic}(\mu_k, s_k)$
        - pdf of logistic distribution: $f(x|\mu, s) = \frac{\exp(-\frac{x-\mu}{s})}{s(1+\exp(-\frac{x-\mu}{s}))^2}$
        - cdf of logistic distribution: $F(x|\mu, s) = \sigma(\frac{x-\mu}{s}) = \frac{1}{1+\exp(-\frac{x-\mu}{s})}$
    - $P(x|\pi, \mu, s) = \sum_{i=k}^{K} \pi_k P(x - 0.5 < \nu < x + 0.5|\mu_k, s_k)$ $= \sum_{k=1}^{K} \pi_k (\sigma(\frac{x+0.5-\mu_k}{s_k}) - \sigma(\frac{x-0.5-\mu_k}{s_k}))$
        - For the edge case of 0, replace $x - 0.5$ by $-\infty$
        - For the edge case of 255, replace $x + 0.5$ by ∞
        - Edge value is assigned with higher probability naturally, which corresponds the observed data distribution.
- Simplify the conditioning of sub-pixels:
    - Original PixelCNN factorizes the model over 3 sub-pixels, making the model unnecessarily complicated.
    - Condition only on whole pixels in the context $x_{<i}$, and output joint distribution over all 3 sub-pixels
        - $\mu_{i,r}$ depends on $x_{<i}$
        - $\mu_{i,g}$ depends on $x_{<i}$ and linearly depends on $x_{i,r}$
        - $\mu_{i,b}$ depends on $x_{<i}$ and linearly depends on $x_{i,r}$ and $x_{i,b}$
        - $s_{i,r}, s_{i,g}, s_{i,b}$ only depends on $x_{<i}$
- Add short-cut connections:
    - The downsampling and upsampling process loses information. Adding long-range skip connections helps recovering this information.
    - Network structure:


32x32
16x16
8x8
8x8
16x16
32x32
x
x
= Sequence of 6 layers
= Downward stream
= Downward and rightward stream
= Identity (skip) connection
= Convolutional connection

- Add dropout as regularization to avoid overfitting.

Experiments

Pix2pix

- Discriminator:
  - Pixel Discriminator
  - PatchGAN Discriminator
- Generator
  - Unet Generator
- Settings
  - batch size: 100
  - num of generator filters in the first conv layer: 64
  - num of discriminator filters in the first conv layer: 64
  - optimizer: Adam with lr = 0.0002

Pixel + Unet

- Loss
  - D_real: 0.6152
  - D_fake: 0.6087
  - G_GAN: 0.9055
  - G_L1: 17.0962

(pic-pixelUnet)

PatchGAN + Unet

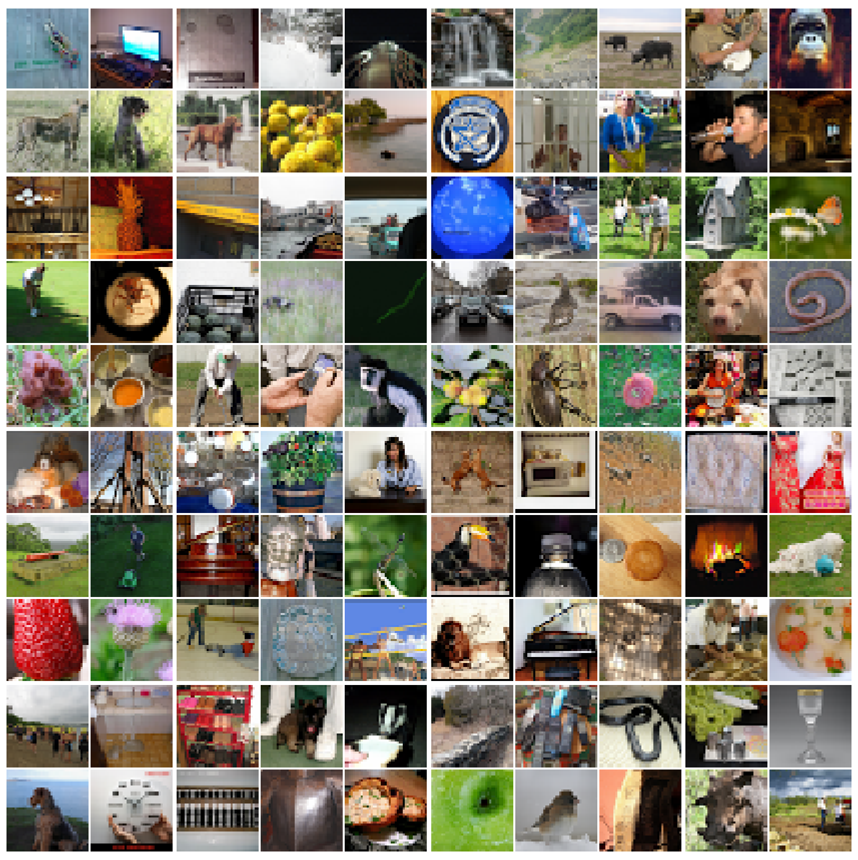
- Loss
  - D_real: 0.0347
  - D_fake: 0.0355
  - G_GAN: 3.9119
  - G_L1: 16.5434

(pic-patchganUnet)

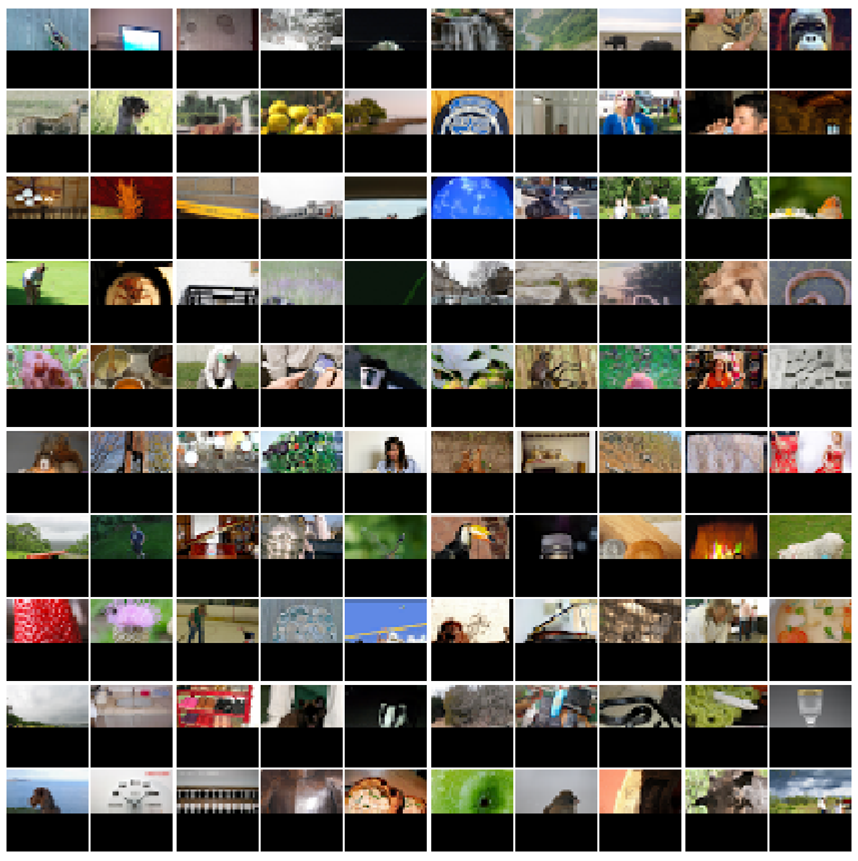
Pixel CNN++

- Experiment settings:
  - number of filters = 160 for all convolution layers
  - number of logistic mixtures K = 10
  - number of residual blocks per stage of the model = 5
  - dropout = 0.5
  - batch_size = 12
  - optimizer = Adam(lr=0.001, lr_decay=0.999995)
- Training on a machine with one 1080Ti GPU achieves 3.93 bits per dimension on test data in about two days.
- Generated samples:

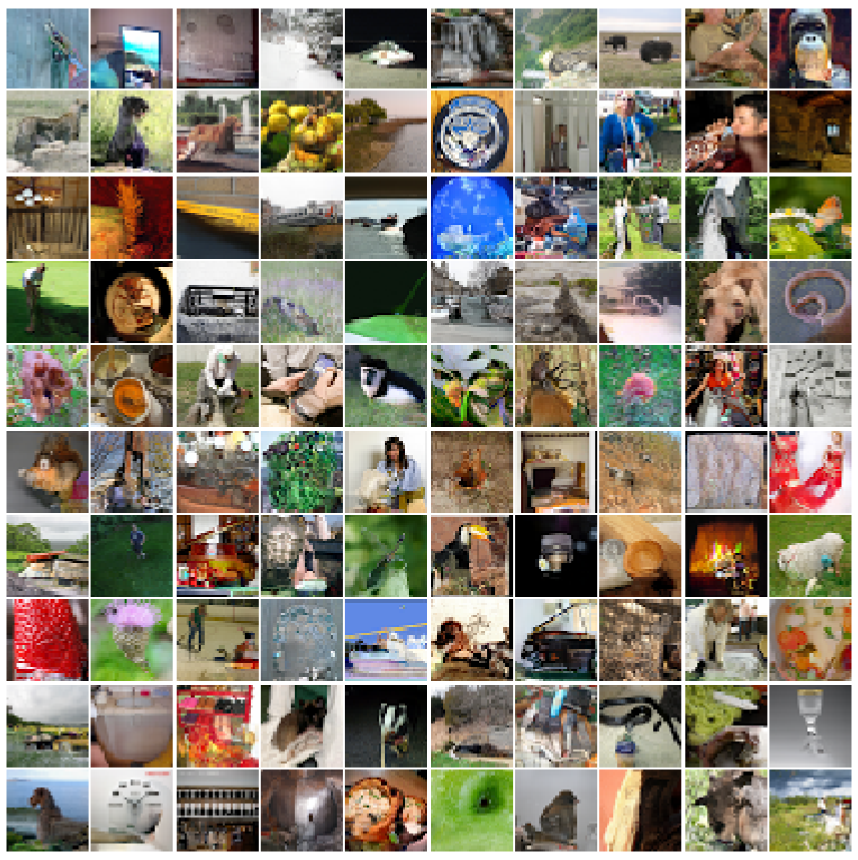
- Test data:



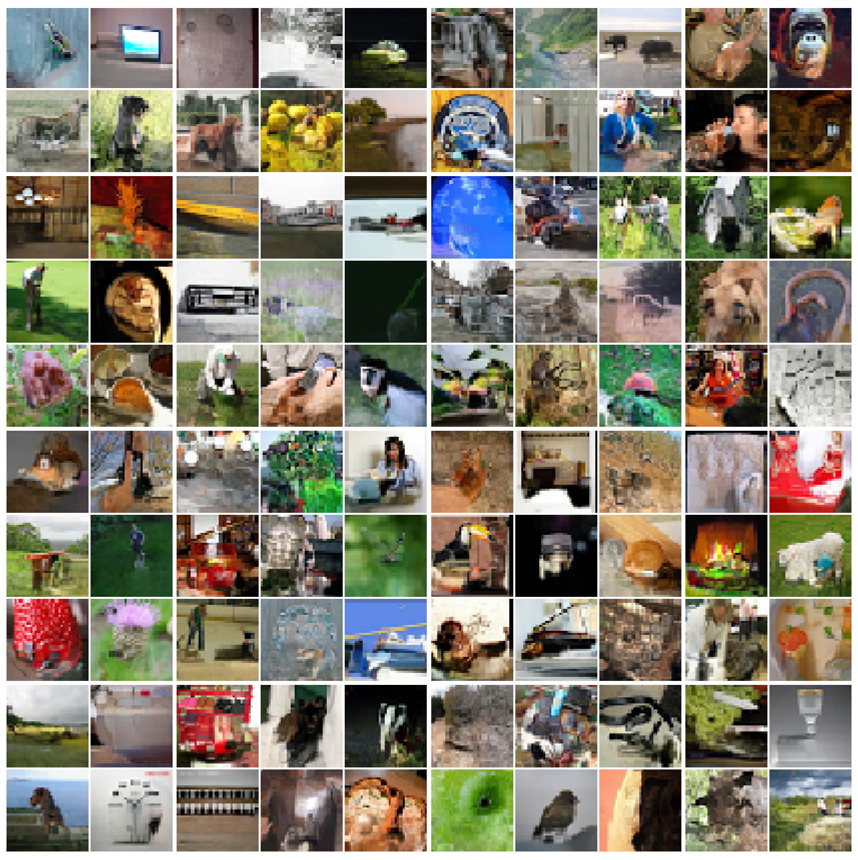
- Input (remove the lower half of each image):



- Samples generated by our model:



- Samples generated by the pre-trained model that achieves state-of-the-art 2.92 bits per dimension:



Conclusion

- We tried a conditional GAN model Pix2Pix to impaint broken images. However, the generator only sees upper half of the image as condition, which is complex in dimentionality and sometimes insufficient. Therefore, we encounter some mode collapse problem, and the generated images are blurred.
- As for the PixelCNN++ method, although the bpd loss of our model is higher compared to the pre-trained model, the generated images are perceptually reasonable. In general, it is hard to completely recover the lost part of the image, even using the state-of-the-art pre-trained model. Nevertheless, this generation method can capture the overall object and its shape, generating highly convincing image from broken one.

Future work

References

- Paper: Image-to-Image Translation with Conditional Adversarial Networks (https://arxiv.org/pdf/1611.07004.pdf)
- Paper: Pixel Recurrent Neural Networks (https://arxiv.org/pdf/1601.06759.pdf)
- Paper: PixelCNN++: A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications (https://arxiv.org/pdf/1701.05517.pdf)
- Github: PyTorch implementaion of image-to-image translation (https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix)
- Github: Tensorflow implementation of PixelRNN (https://github.com/carpedm20/pixel-rnn-tensorflow)
- Github: Tensorflow implementation of PixelCNN++ (https://github.com/openai/pixel-cnn)

Environment

- Linux x86_64 16.04.2-Ubuntu
- python: 3.5.2
- tensorflow: 1.4.1
- pytorch version need to be 0.3.0.post4 to enable scheduler
  - pip3 install http://download.pytorch.org/whl/cu80/torch-0.3.0.post4-cp35-cp35m-linux_x86_64.whl (http://download.pytorch.org/whl/cu80/torch-0.3.0.post4-cp35-cp35m-linux_x86_64.whl) --user
  - pip3 install --no-deps torchvision
- pillow: 4.3.0
- visdom: 0.1.7
- dominate: 2.3.1
- imageio: 2.2.0