Neural Networks and Deep Learning

Generative Adversarial Network

Shumin Wu

Department of Computer Science
shumin@colorado.edu

April 8, 2020

University of Colorado Boulder

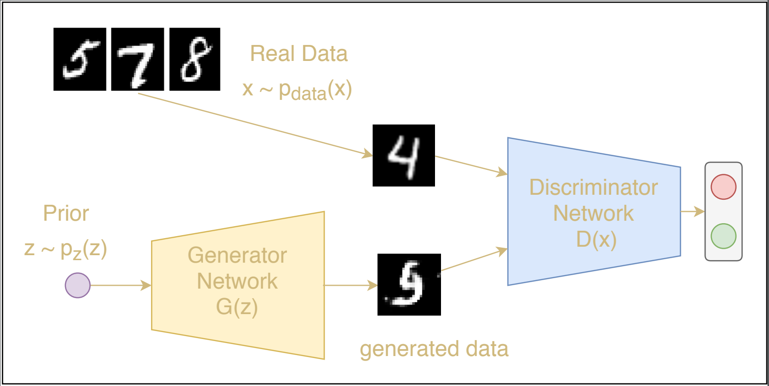
A GAN is a zero-sum game between two adversaries, a generator (G) and a discriminator (D).

G generates samples from a learned distribution p_G and tries to trick D into believing they are from p_{data}, the true data distribution.

D tries not to be deceived.

	Generator	Discriminator
Input	A random vector	A sample from p_G or p_{data}
Output	Sample generated from p_G	Probability that input $\sim p_{data}$
Task	Make p_G close to p_{data}	Distinguish between p_G and p_{data}

G and D are neural networks – typically, though not necessarily, ConvNets.



Generative Adversarial Networks Mark Chang

Within a training iteration, repeat the following k times to optimize the weights of the discriminator D.

Given

- a minibatch of m noise samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$, and
- a minibatch of m examples from $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$

update D with gradient *ascent*:

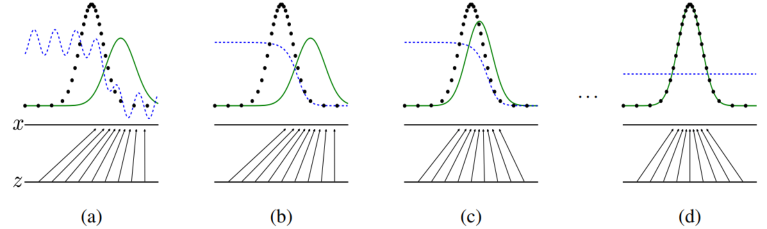
$$\nabla_{d_\theta} \frac{1}{m} \sum_{i=1}^{m} \left[logD\left(\mathbf{x}^{(\mathbf{i})}\right) + log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)\right].$$

Within a training iteration, do the following *once* to optimize the weights of the generator G.

Given a minibatch of m noise samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$, update G with gradient *descent*:

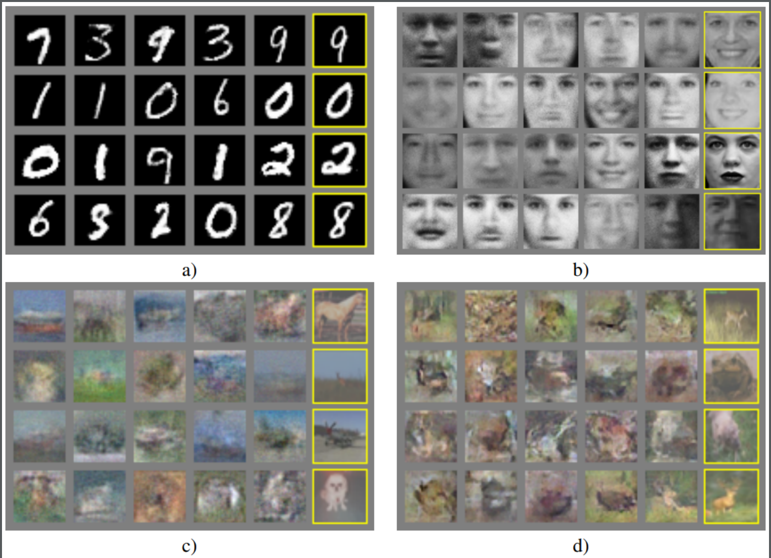
$$\nabla_{g\theta} \frac{1}{m} \sum_{i=1}^{m} log\Big(1 - D\Big(G\Big(\mathbf{z}^{(i)}\Big)\Big)\Big).$$

$$min_G max_D V(D, G) = \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{z \sim P_{noise}}[log(1 - D(G(z)))]$$



Training process for a GAN

discriminating distribution - blue, generating distribution - green, data generating distribution - black. Source: Goodfellow et al, 2014


a)
b)
c)
d)

- Non-convergence - in the zero-sum game played by the generator and discriminator, the equilibrium can be evasive. The progress made by one player may, in turn and repeatedly, be undone by the other player.
- Mode collapse, mode dropping. Real data are multimodal. Mode collapse occurs when the generator settles into a state where it outputs samples from one or a small number of modes. The effect is that the generator creates samples that are far less diverse than those found in the real data.

Goodfellow, 2016

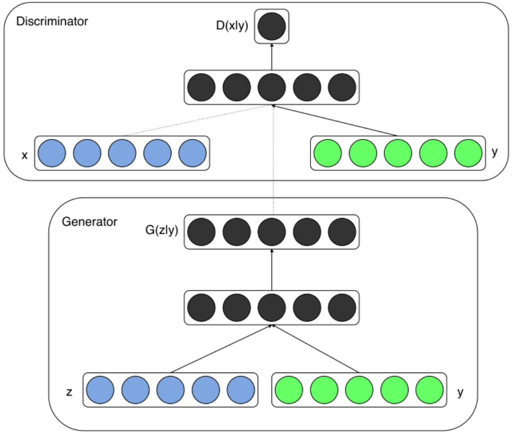
It is common to add noise during training of generative models.

Manifolds of p_{data} and p_g

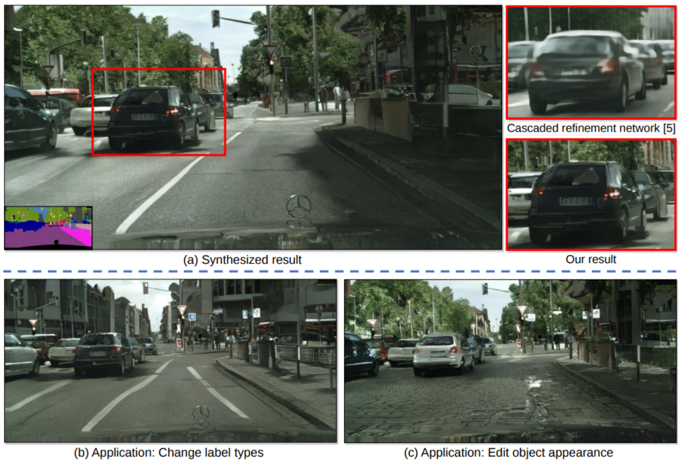
Manifolds of p_{data} and p_g with noise

- Eliminates lack of common support between p_{data} and p_g.
- Makes D perform worse (initially), so gradients of D are non-zero
- Ensures that KL-divergence is defined and the GAN convergence proof holds (modulo comment at end of original GAN proof)

See Sonderby et al, 2017

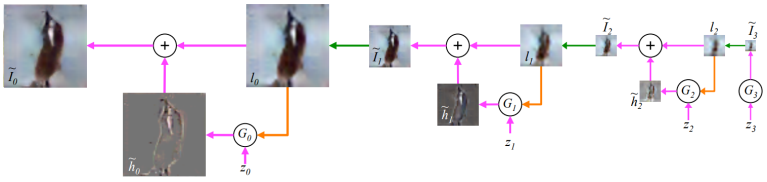


Mizra and Osindero, 2014



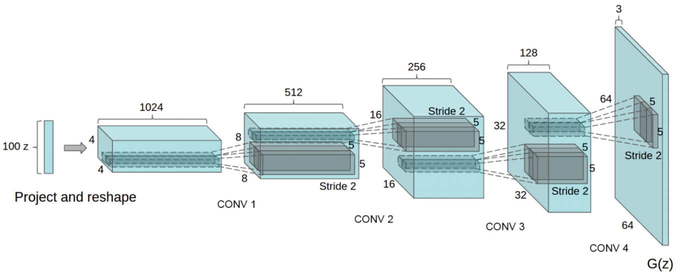
Wang et al, 2018

First flavor of GANs to scale to “high resolution” images (64×64).



Training procedure for LAPGAN Denton et al, 2015

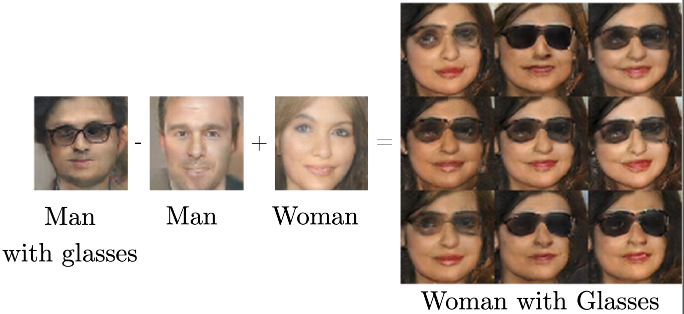
LAPGAN samples & sampling procedure Denton et al, 2015



Generator for deep convolutional generative adversarial network Radford et al, 2015

Increase quality of generator G, by

- adding batch normalization layers to G and D
- optimizing using Adam instead of SGD



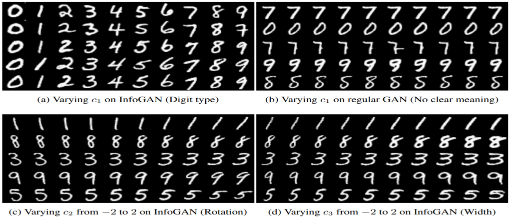
Vector Arithmetic of Z vector Radford et al, 2015

GAN loss

$$min_G max_D V(D, G) = \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{z \sim P_{noise}}[log(1 - D(G(z)))]$$

InfoGAN loss

$$min_G max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$$



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)

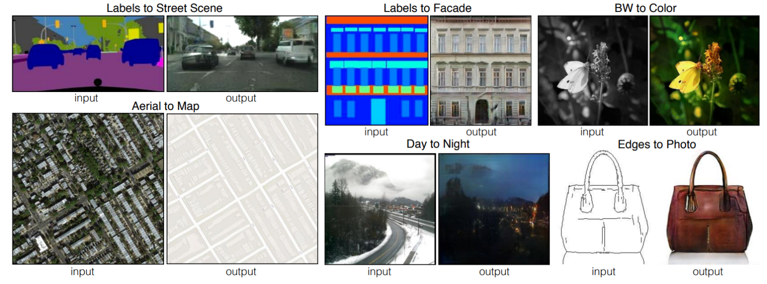
(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Chen et al, 2016

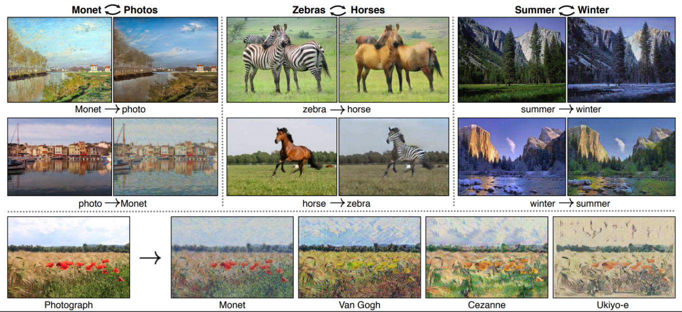
Image-to-image translation using conditional GANs and paired images.

An online demo illustrates the basic approach well – particularly the building facades.



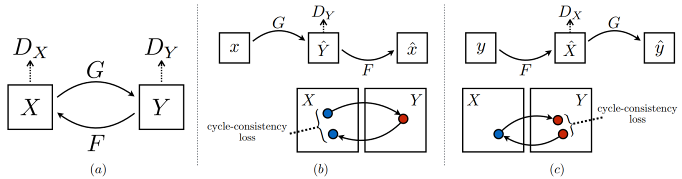
Isola et al, 2016

In CycleGAN, the authors use the GAN framework and corpora of unpaired images to learn to translate salient features between domains.



Zhu et al, 2017

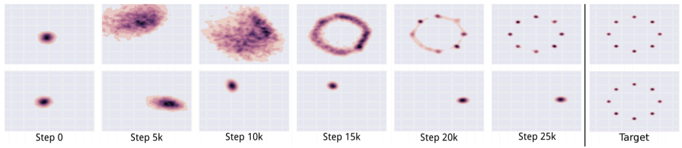
The pair of generators G and F map between domains (i.e. $G: X \rightarrow Y$ and $F: Y \rightarrow X$).



Zhu et al, 2017

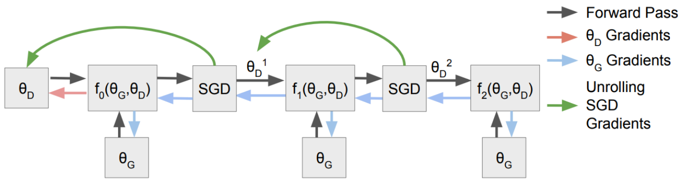
The study showed by ablation that a cycle consistency loss that ensured $F(G(X)) \sim X$ and $(G(F(Y)) \sim Y$ substantially improved the quality of generated images.

Peek into the future of the generator/discriminator game to learn the long term utility of an update to the generator.



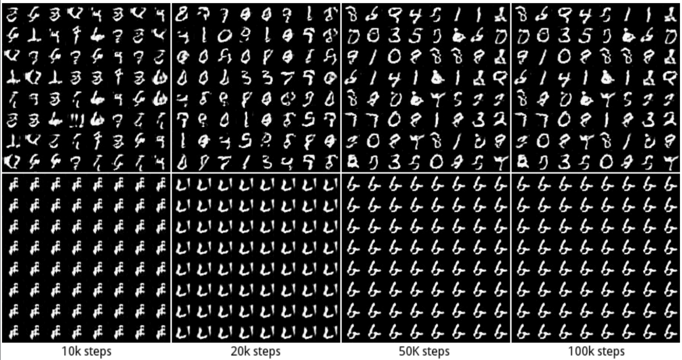
Metz et al, 2016

Generator updated from gradients of multiple time steps.
Discriminator updated with gradients of a single step.



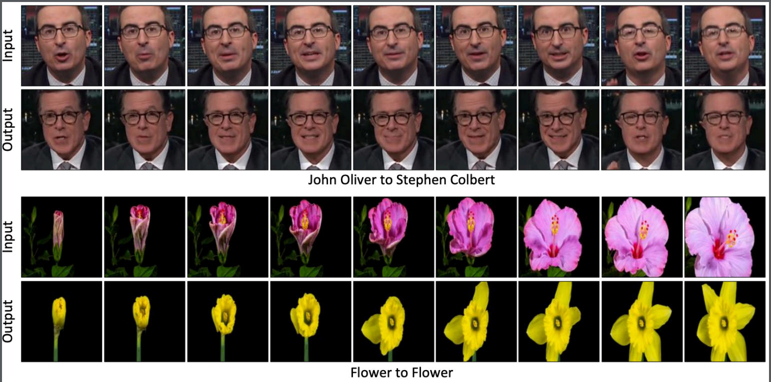
Metz et al, 2016

RNN-based GAN with and without unrolling



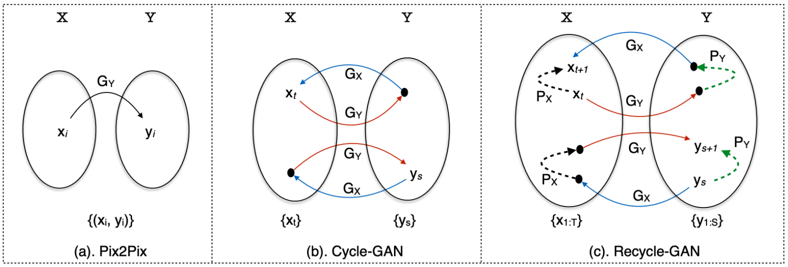
Metz et al, 2016

In RecycleGAN, the authors adds temporal constraints to CycleGAN to improve video retargeting.

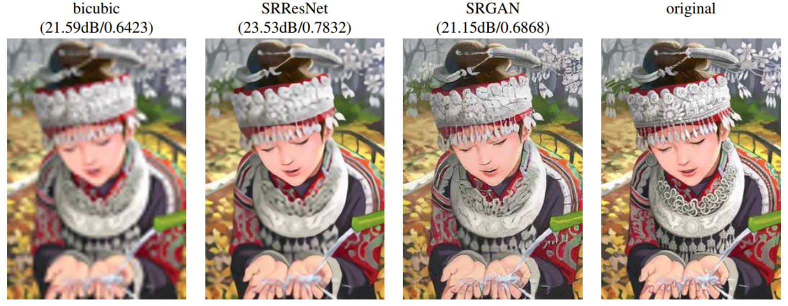


Demo

Video retargeting by training next frame predictors P_X & P_Y.



Bansal et al, 2018

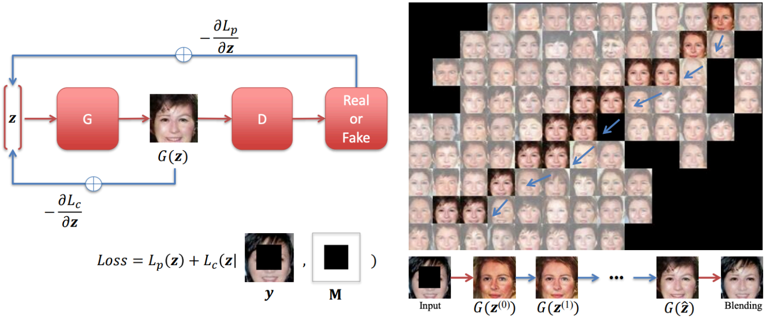


Ledig et al, 2017

Some methods for measuring generator quality. See Lucic et al, 2018.

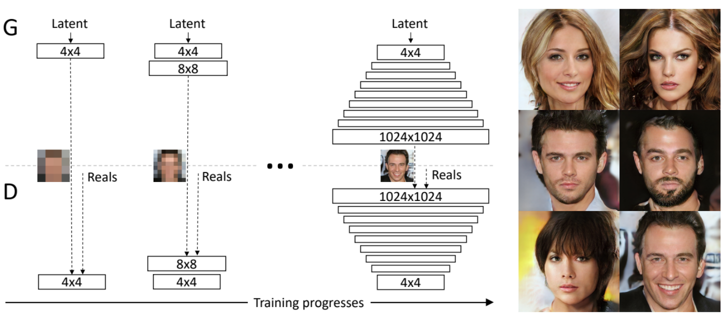
- *Inception Score* takes into account the entropy of the distribution of labels (i.e. softmax output) of generated samples ($p(y|x)$) and the variance of the classes using an Inception model trained on ImageNet.
- *Fréchet Inception Distance* is the Fréchet distance between two multivariate Gaussians, $\mathcal{N}(\mu_x, \Sigma_x)$ and $\mathcal{N}(\mu_g, \Sigma_g)$, where the parameters of the distributions are estimates from the Inception embeddings of the real and generated data.

Train a GAN, encode the masked input into latent code z and generate the missing region.

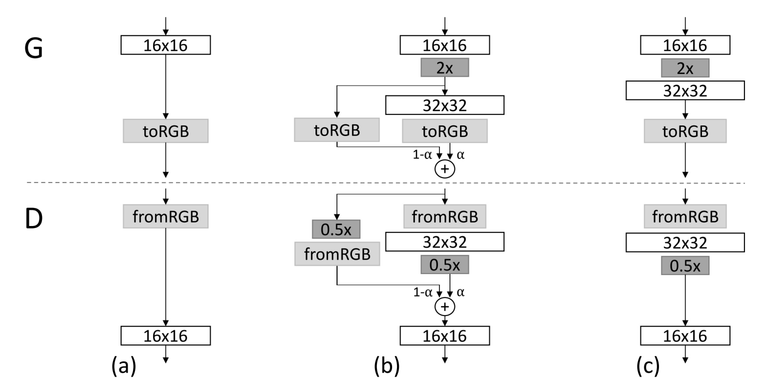


Yeh et al, 2016

Progressively adding layers of the generator and discriminator allows scaling up to images of size 1024×1024.

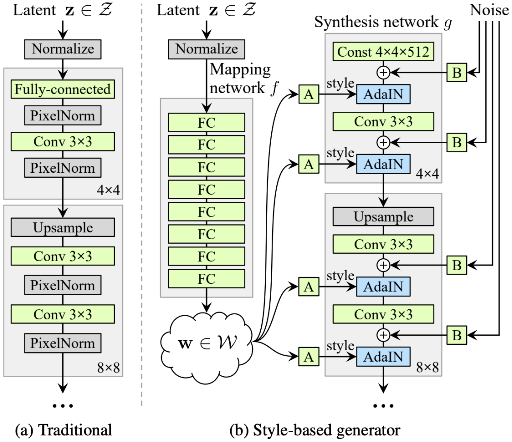


Training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Karras et al, 2017



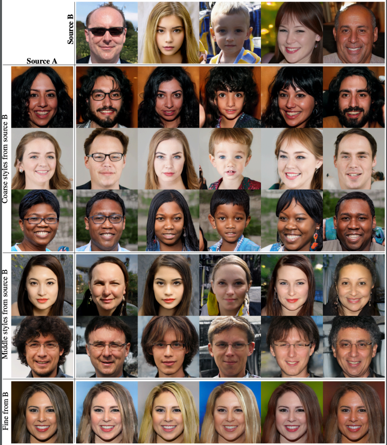
Karras et al, 2017

Style-based image generation.



Karras et al, 2018

Style transfer by substituting latent z of a second image for certain layers of a progressive GAN.

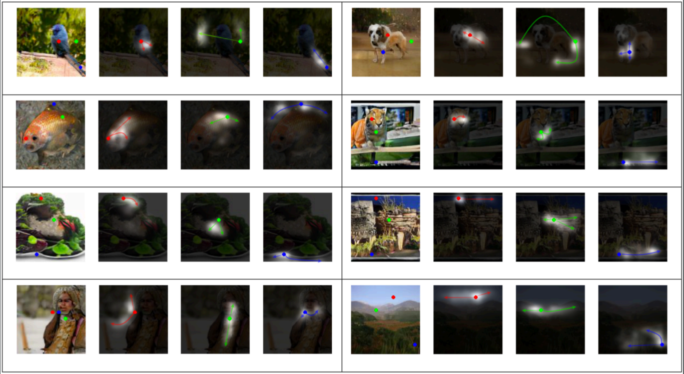


Coarse to fine style transfer. Karras et al, 2018

Progressive growth phase artifact

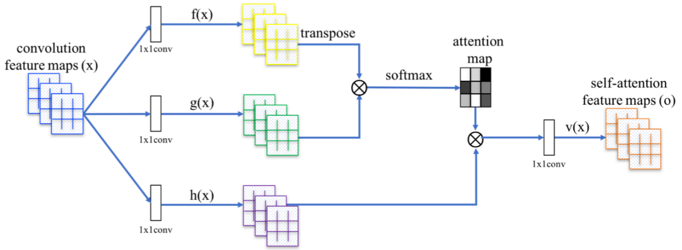
Teeth does not follow pose. Karras et al, 2019

GAN with self-attention.



Zhang et al, 2019

Dot-product self-attention of CNN feature maps (with channel/filter size compression).



Zhang et al, 2019