Auto-Encoders and GANs for Computer Vision

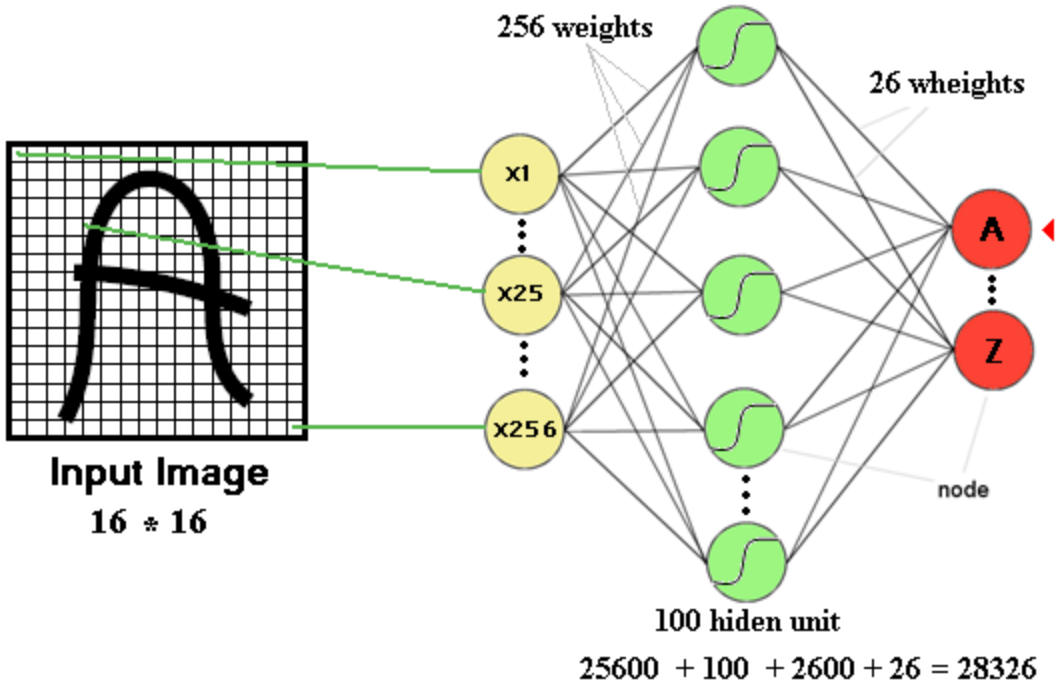
By

Dr. Subrahmanyam Murala
CVPR Lab
Department of Electrical Engineering
IIT Ropar, Rupnagar, Punjab

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

- The number of trainable parameters becomes extremely large.

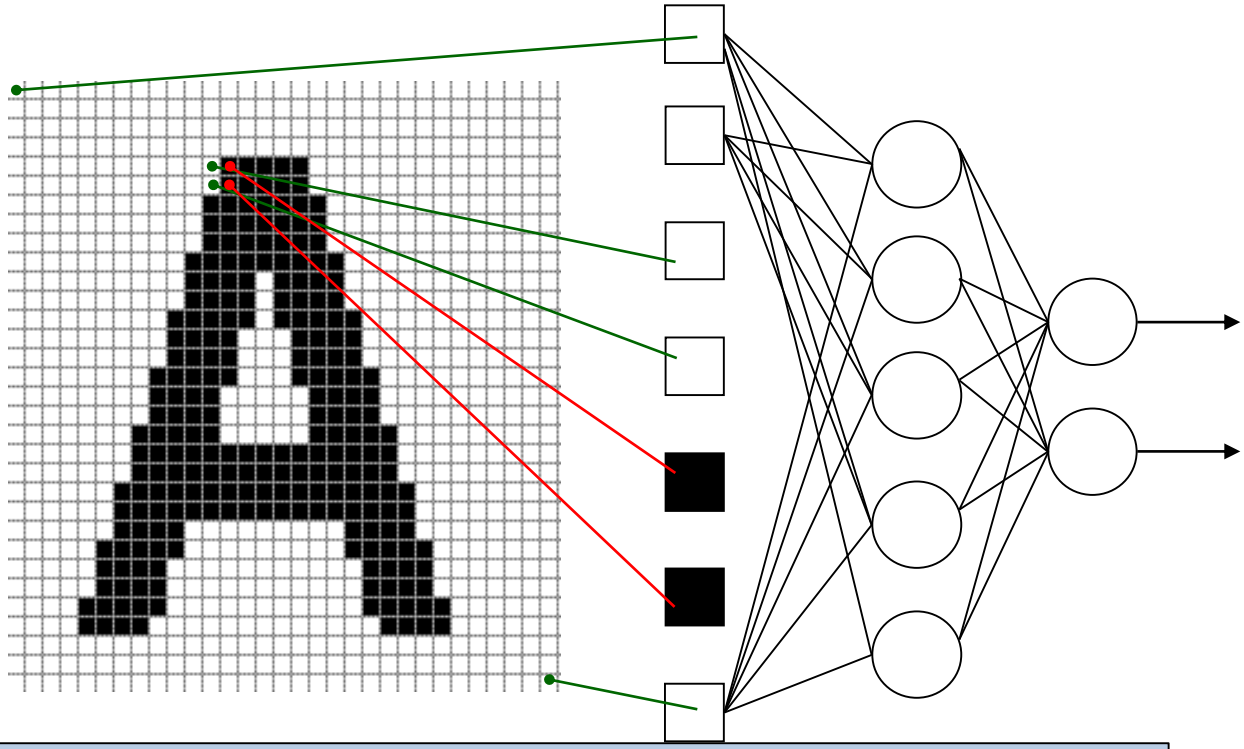


Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

- Little or no invariance to shifting, scaling, and other forms of distortion

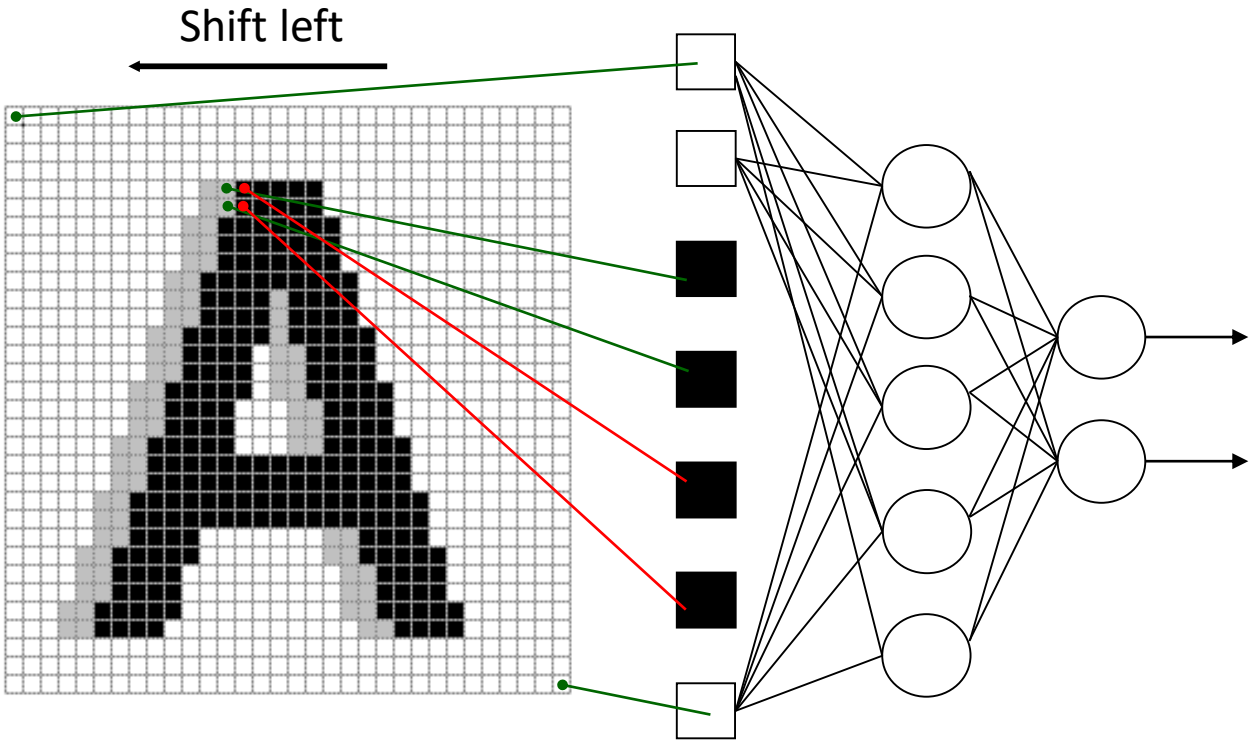


Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

- Little or no invariance to shifting, scaling, and other forms of distortion

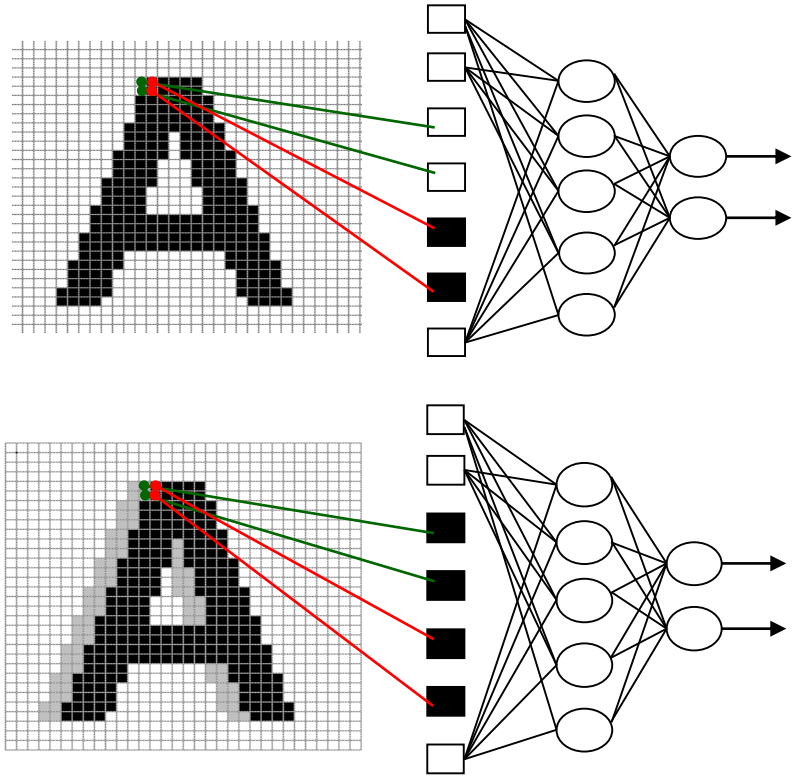


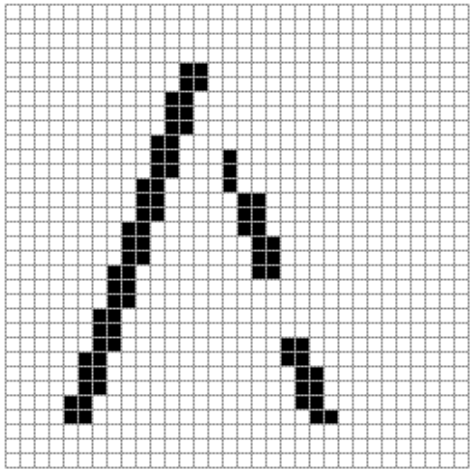
Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

- Little or no invariance to shifting, scaling, and other forms of distortion



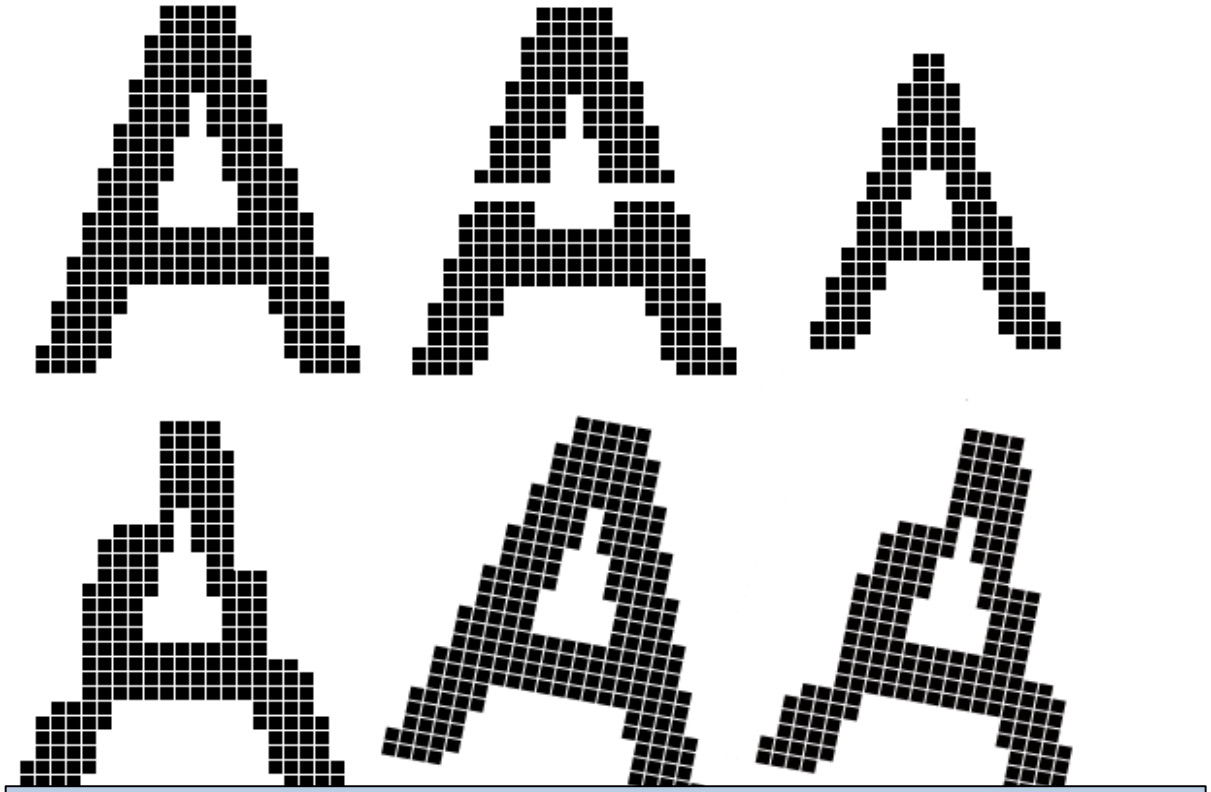


Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

- Little or no invariance to shifting, scaling, and other forms of distortion

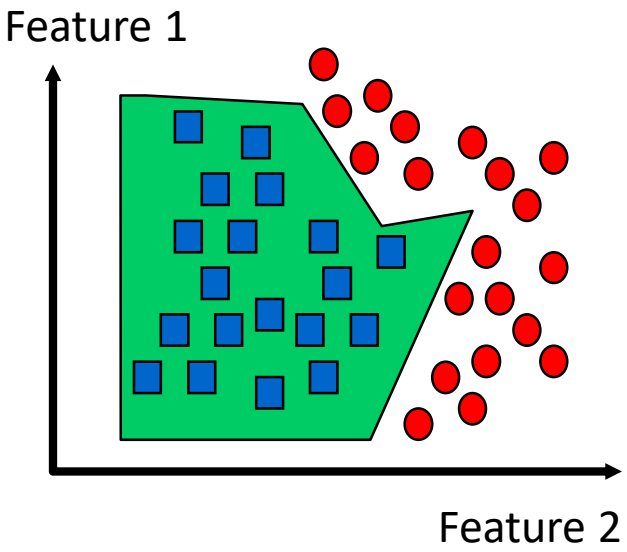


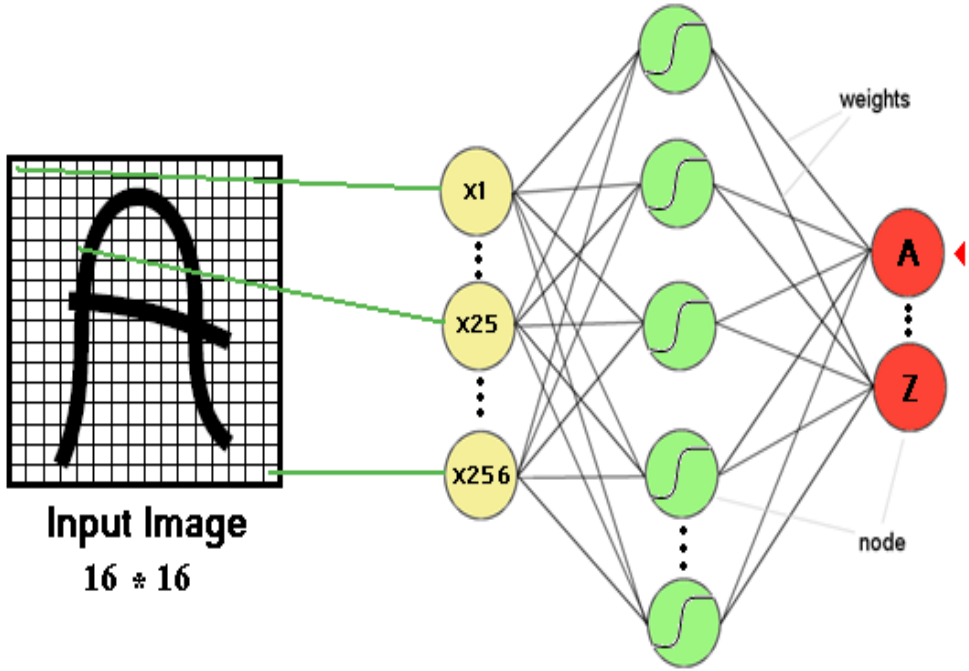
Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

- the topology of the input data is completely ignored
- work with raw data.





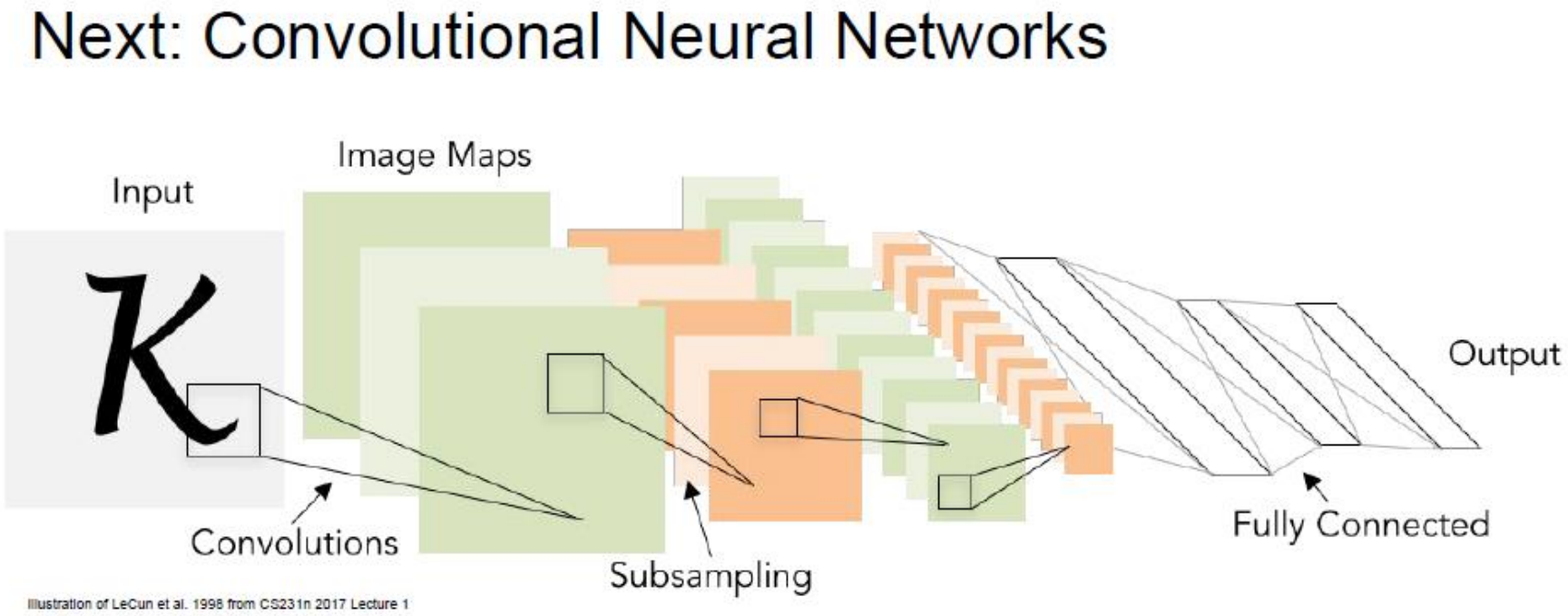
Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Drawbacks of Neural Networks

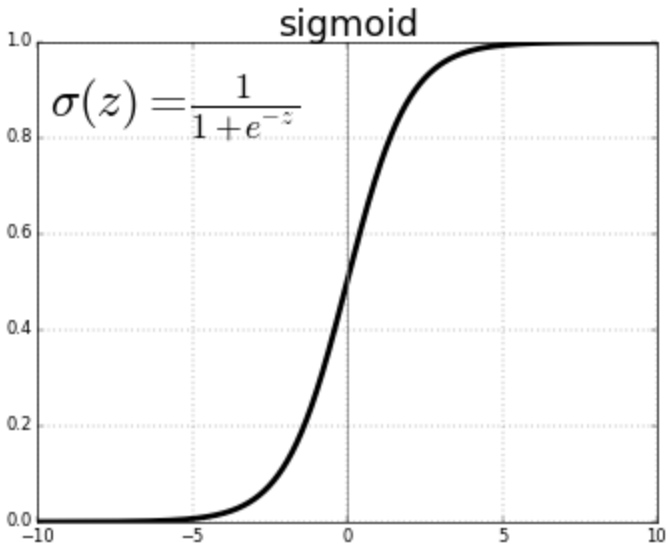
- Feature extraction and training

CNN

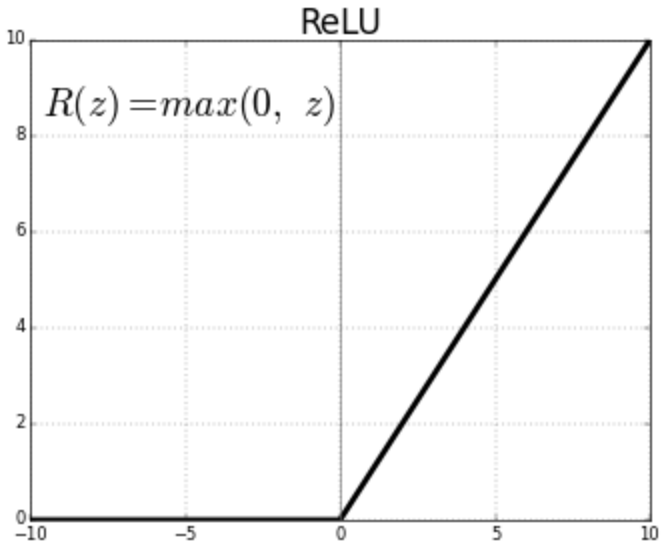


Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

ReLU: Rectified Linear Unit



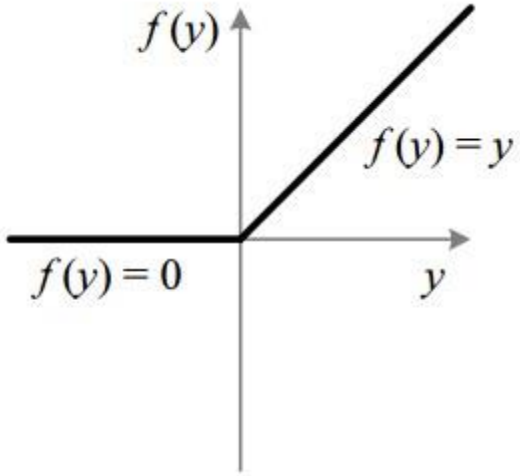
Range: [0 to 1]



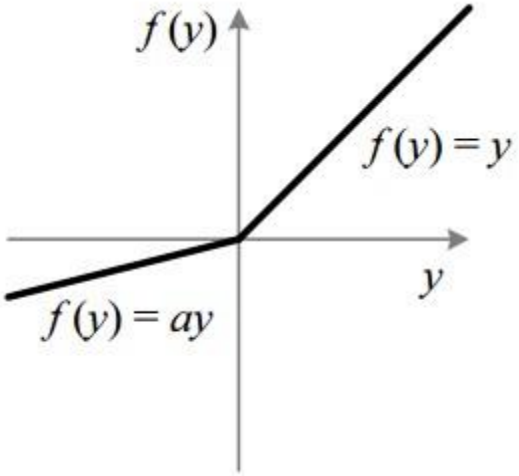
Range: [0 to infinity)

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Leaky ReLU: Leaky Rectified Linear Unit



ReLU



Leaky ReLU

Range: [- infinity to infinity)

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning Terminology

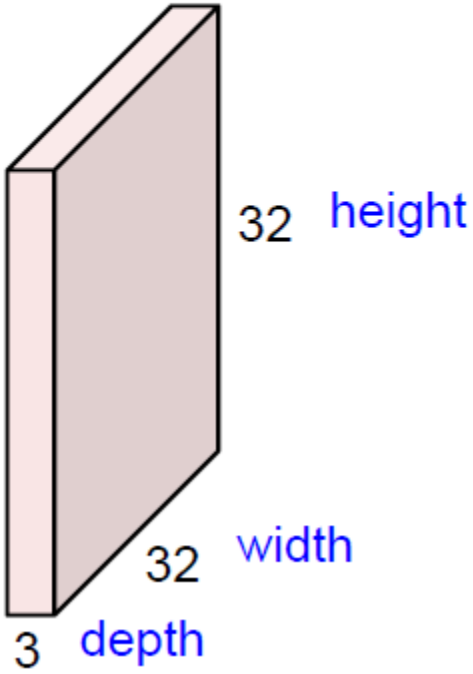
- Epoch: one forward pass and one backward pass of all training examples.
- (Mini) Batch size: Number of training examples in one forward/backward pass. The higher the batch size is, the more memory space you'll need.
- Number of iterations: Number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass
- Tensor: Tensor is an n-dimensional array. Tensors can keep track of a computational graph and gradients, but they're also useful as a generic tool for scientific computing.
- Learning rate: It determines how fast we want to update the weights during optimization, if learning rate is too small, gradient descent can be slow to find the minimum and if it's too large gradient descent may not converge(it can overshoot the minima).
- Data Normalization
- Dropout: Dropout is a regularization technique for reducing overfitting in neural networks. At each training step we randomly drop out (set to zero) set of nodes, thus we create a different model for each training case, all of these models share weights. It's a form of model averaging.

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Convolution Layer

32x32x3 image -> preserve spatial structure



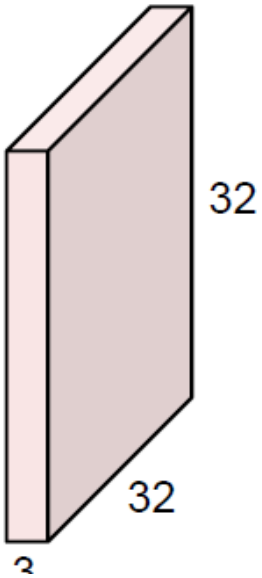
Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

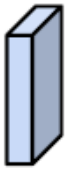
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

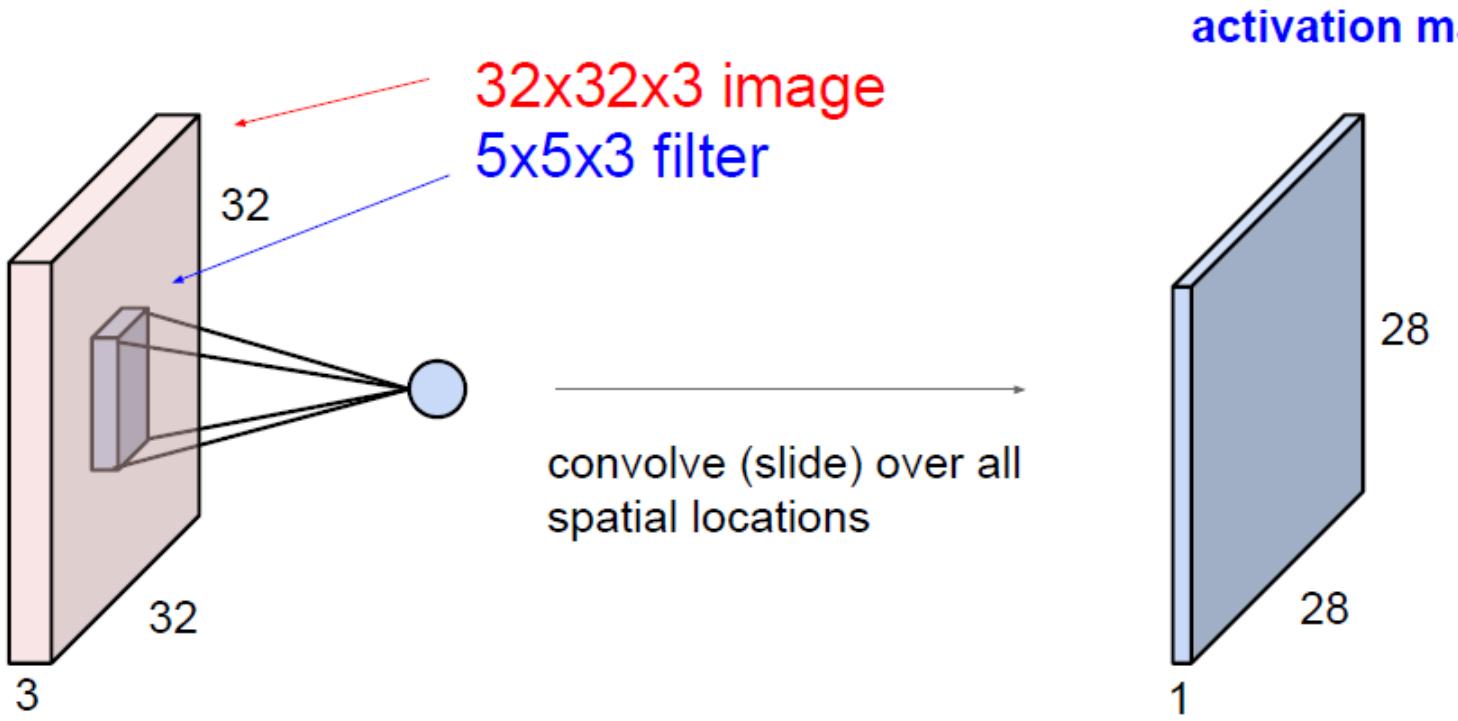


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

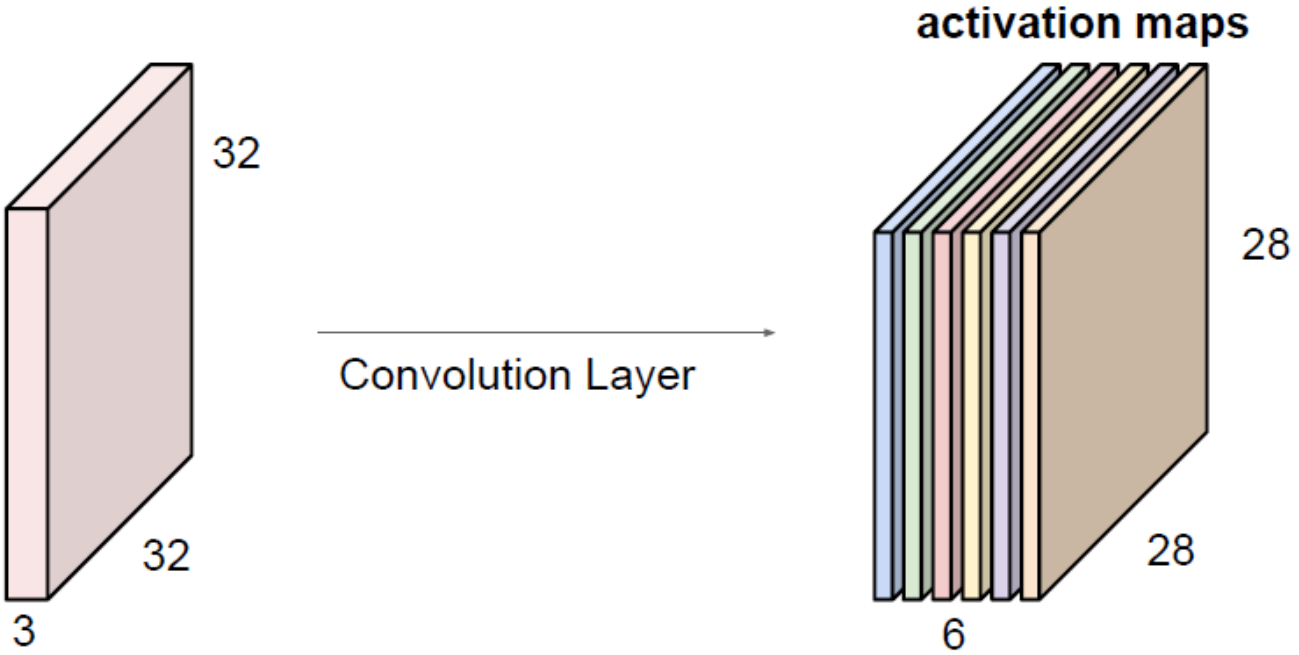
Convolution Layer



Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



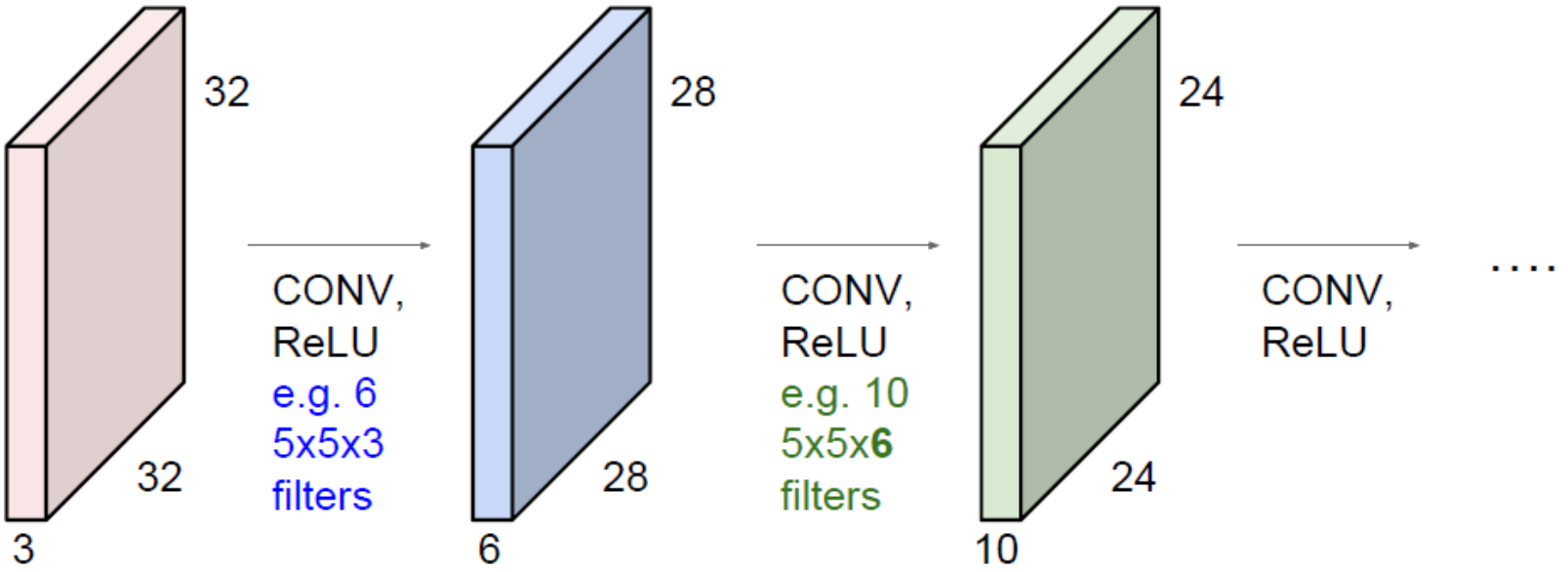
We stack these up to get a "new image" of size 28x28x6!

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.

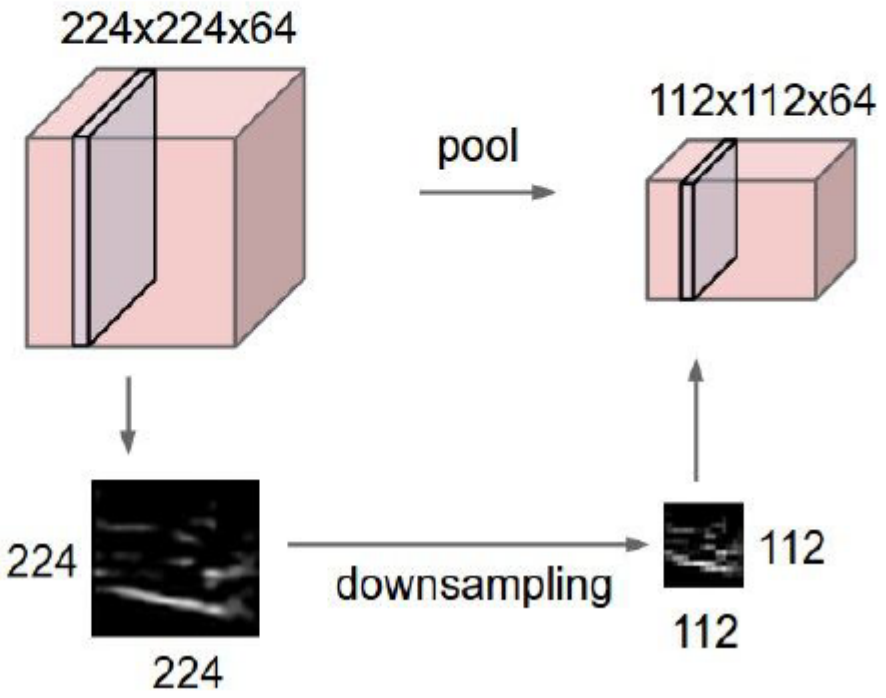


Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Pooling layer

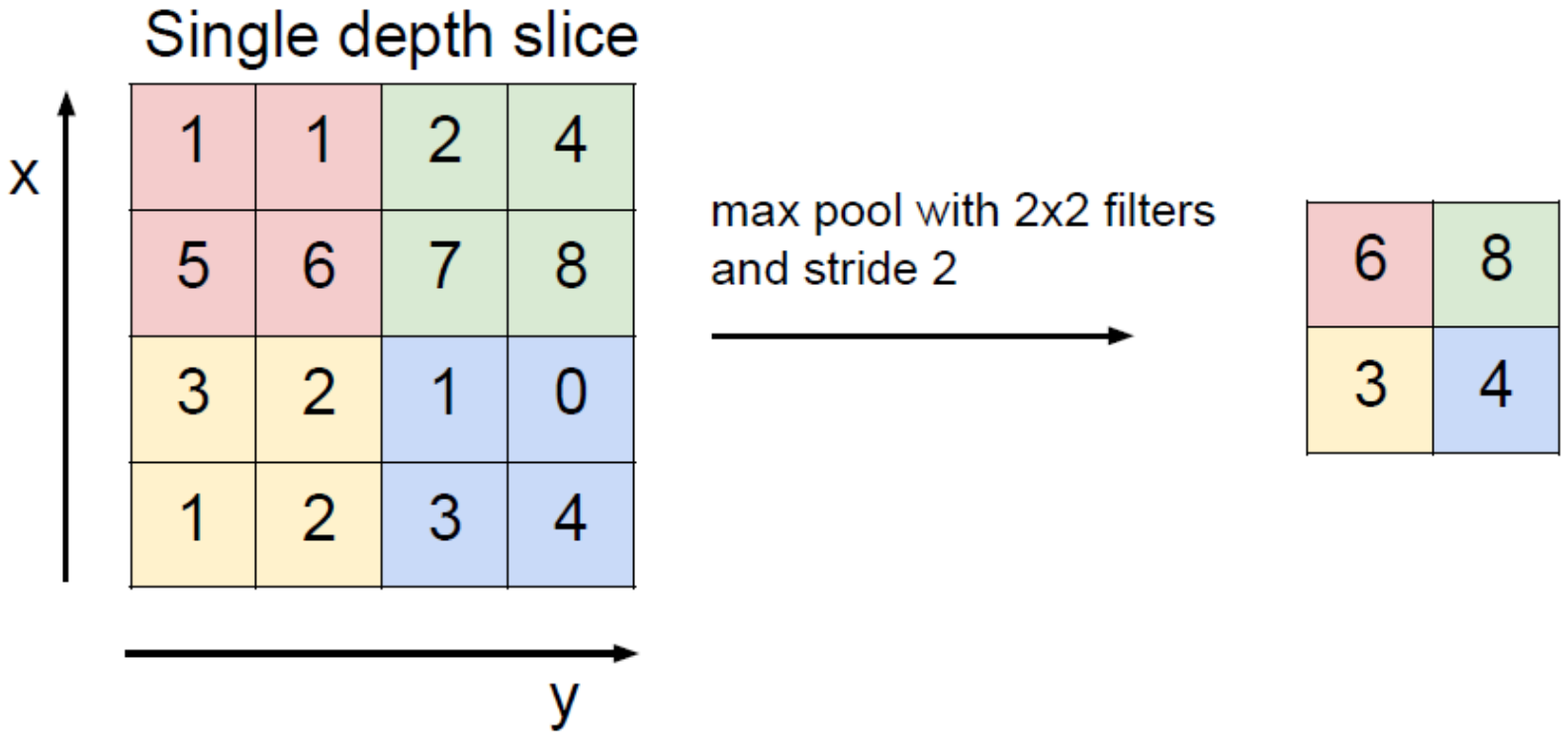
- makes the representations smaller and more manageable
- operates over each activation map independently:



Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

MAX POOLING

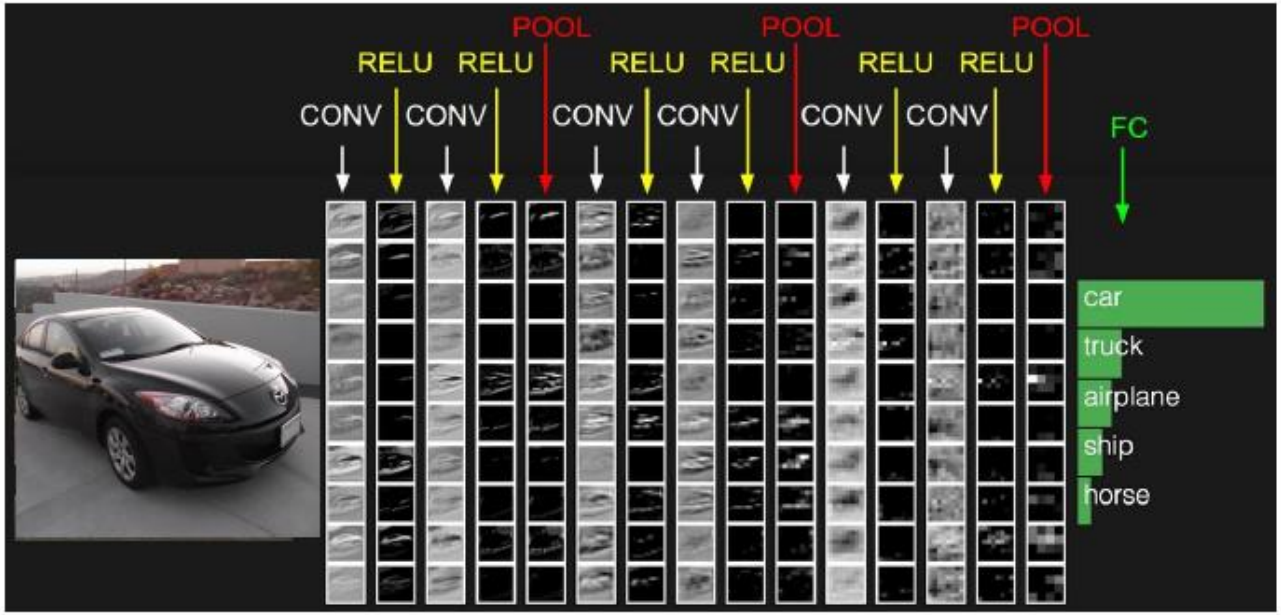


Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

Deep Learning: Convolutional Neural Networks

Fully Connected Layer (FC layer)

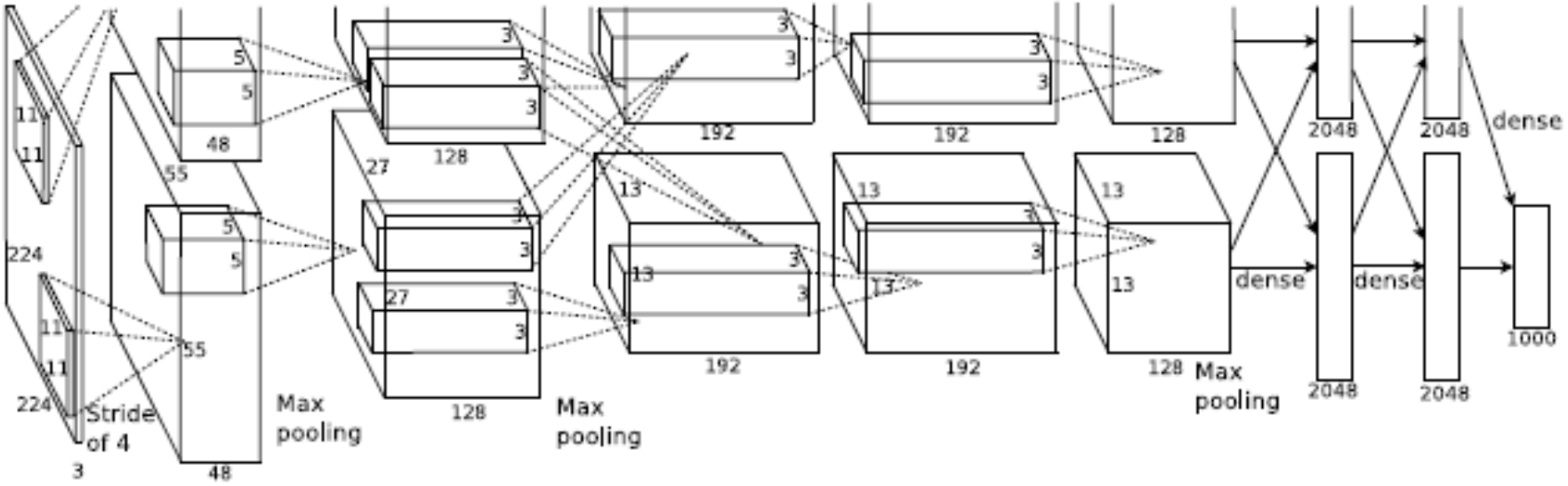
- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

AlexNet

Source: Link

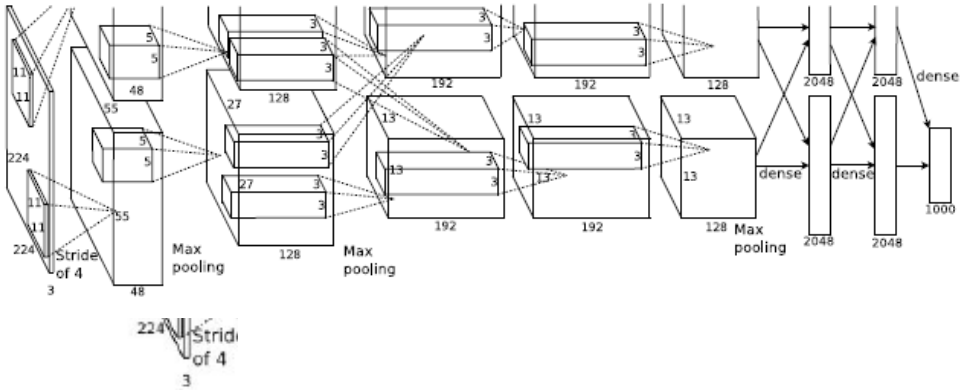


Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

AlexNet



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

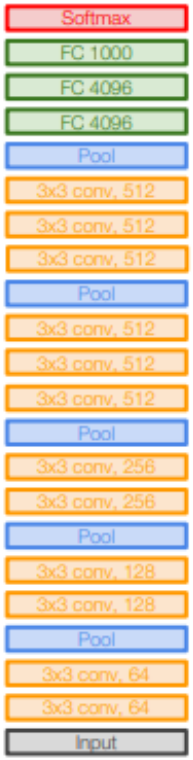
Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

VGG-16

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

VGG-16

INPUT: [224x224x3] memory: 224*224*3=150K params: 0 (not counting biases)
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Note:

Most memory is in early CONV

Most params are in late FC

Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

ResNet 50, 101, 152

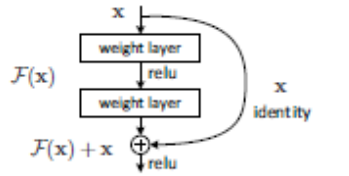
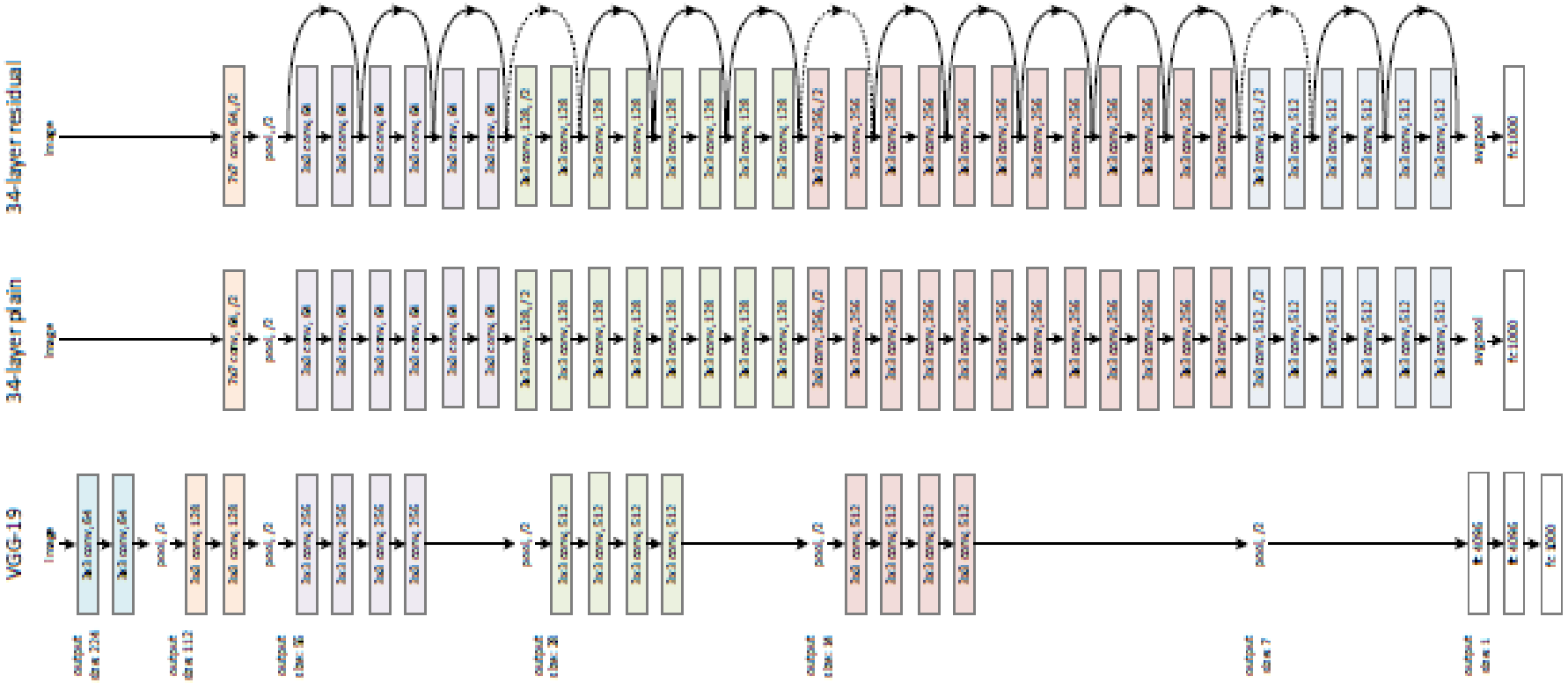
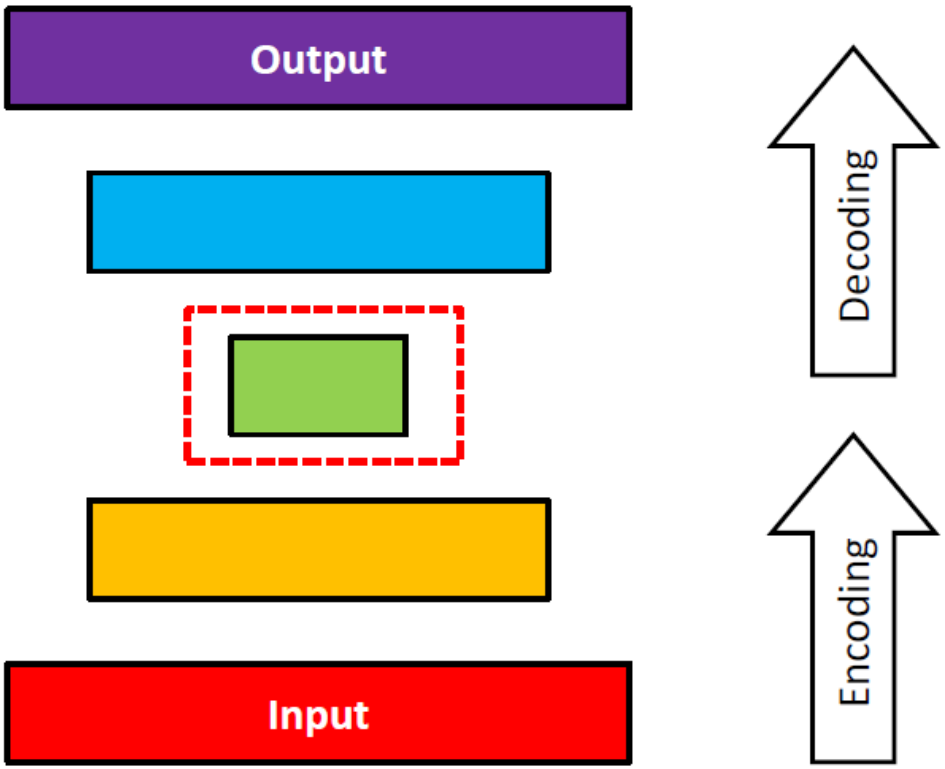


Figure 2. Residual learning: a building block.



Lectures of CS231 by Fei-Fei Li & Justin Johnson & Serena Yeung.
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

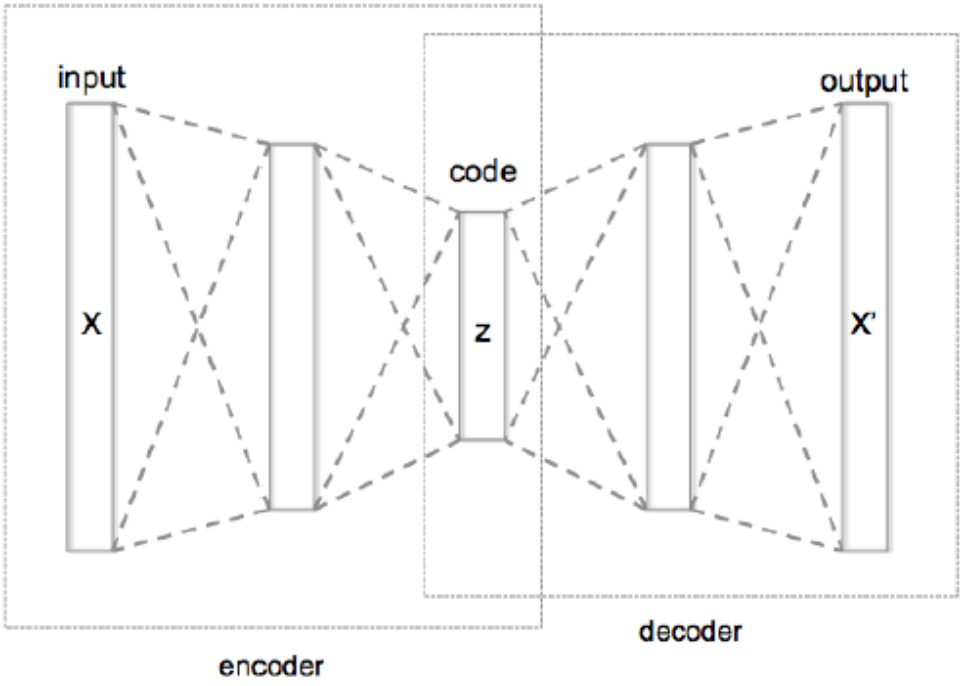
Auto-Encoders



The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

Auto-Encoder?

Encoder + Decoder Structure



The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

Autoencoder

- An autoencoder is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input
- Hidden layer h
- Two parts
  - Encoder h= f(x)
  - Decoder r=g(h)
- Modern autoencoders also generalized to stochastic mappings

$$p_{encoder}(h|x), p_{decoder}(x|h)$$

The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

Autoencoder

- Traditionally, autoencoders were used for dimensionality reduction or feature learning.
- Recently, theoretical connections between autoencoders and latent variable models have brought autoencoders to the forefront of generative modeling
- Autoencoders may be thought of as being a special case of feed forward networks, and may be trained with all of the same techniques, typically minibatch gradient descent following gradients computed by back-propagation

The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

- One way to obtain useful features from the autoencoder is to constrain h to have smaller dimension than x
- Learning an undercomplete representation forces the autoencoder to capture the most salient features of the training data.
- The learning process is described simply as minimizing a loss function

$$L(x, g(f(x)))$$

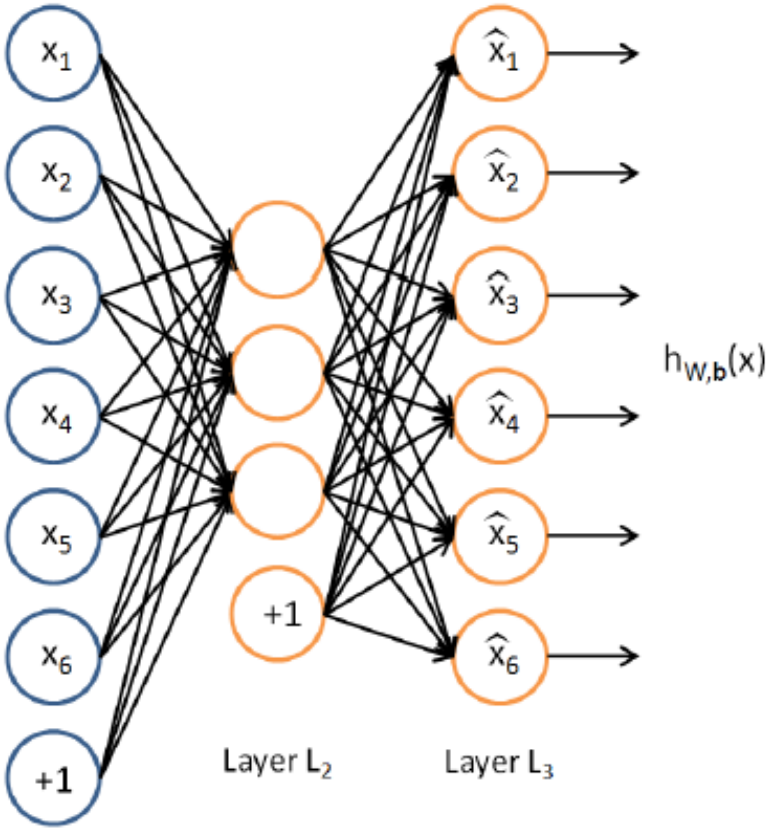
- When the decoder is linear and L is the mean squared error, an undercomplete autoencoder learns to span the same subspace as PCA

The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

- Autoencoders with nonlinear encoder functions f and nonlinear decoder functions g can thus learn a more powerful nonlinear generalization of PCA
- if the encoder and decoder are allowed too much capacity, the autoencoder can learn to perform the copying task without extracting useful information about the distribution of the data

The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

Unsupervised feature learning with a neural network



Autoencoder

Unlabeled training examples set

$\{x^{(1)}, x^{(2)}, x^{(3)} \ldots\}, \; x^{(i)} \in \mathbb{R}^n$

An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. $y^{(i)} = x^{(i)}$

Network is trained to output the input (learn identify function).

$$h_{w,b}(x) \approx x$$

Solution may be trivial.

The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

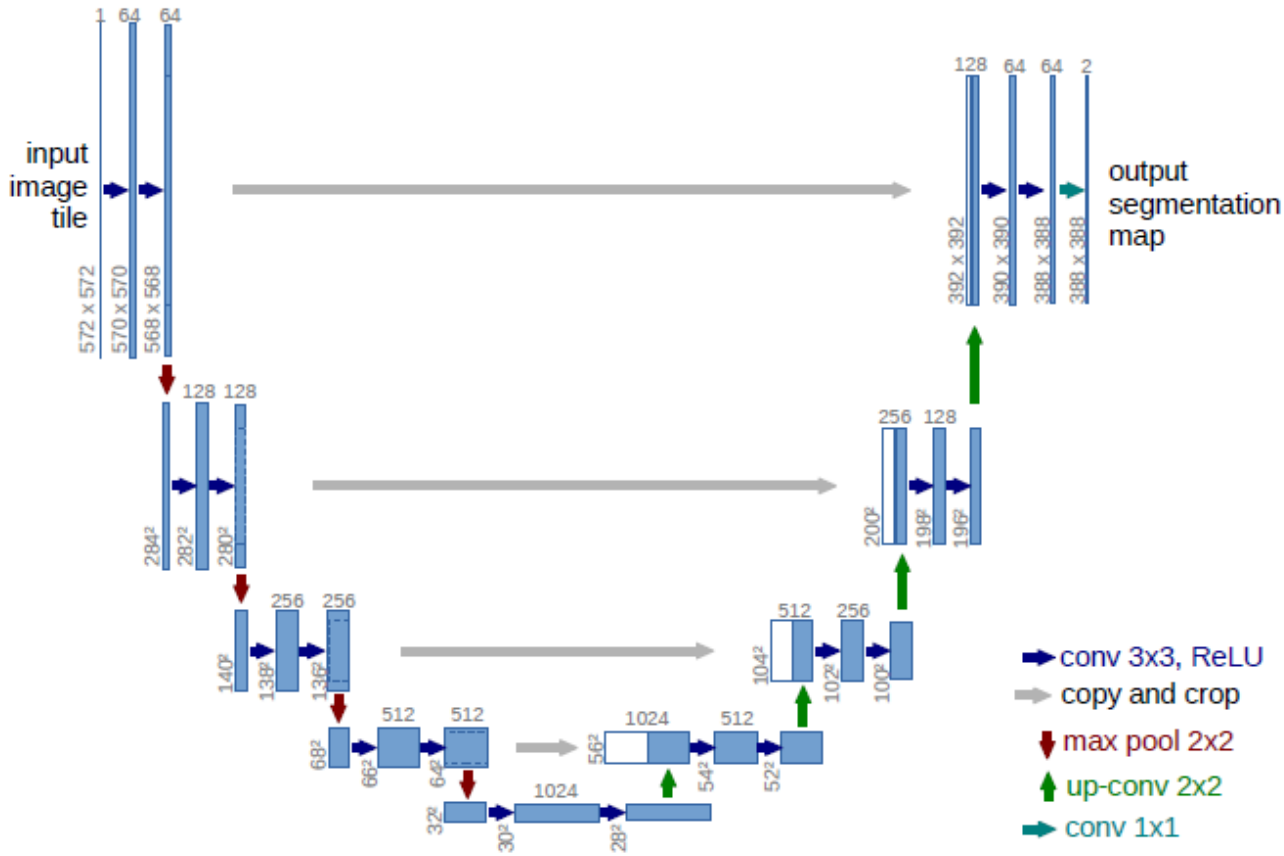
Denoising Auto-Encoder?

Input and output data of the auto-encoder is basically **identical**.

Then, how can we make this network to have **denoising power**?

The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

UNet for Image Segmentation



The content of the Auto-encoders is collected from:
http://cse.iitkgp.ac.in/~sudeshna/courses/DL17/Autoencoder-15-Mar-17.pdf

MSFgNet: A Novel Compact End-to-End Deep Network for Moving Object Detection

Patil Prashant W. and Subrahmanyam Murala, "MSFgNet: A Novel Compact End-to-End Deep Network for Moving Object Detection." *IEEE Transactions on Intelligent Transportation Systems* ***(2018)*** **(Accepted) (Impact Factor: 5.7)**

Main Contributions

In this work, we have proposed very compact encoder-decoder network named as foreground network for moving object detection in videos. The main contributions of the proposed work are given as follows:

- An end-to-end network (MSFgNet) with sub-modules of motion saliency network (MSNet) and foreground network (FgNet) is proposed for moving object detection in videos.
- The network is trained end-to-end with sub-modules of MSNet and FgNet. To the best of our knowledge, it was the first method with end-to-end training for moving object detection in videos.
- MSNet is proposed to estimate the background for a given small video stream as input video. Then, the temporal saliency maps are collected using estimated background and input frames.
- FgNet is proposed with a compact EDnet for foreground extraction on collected temporal saliency maps.

Proposed system framework

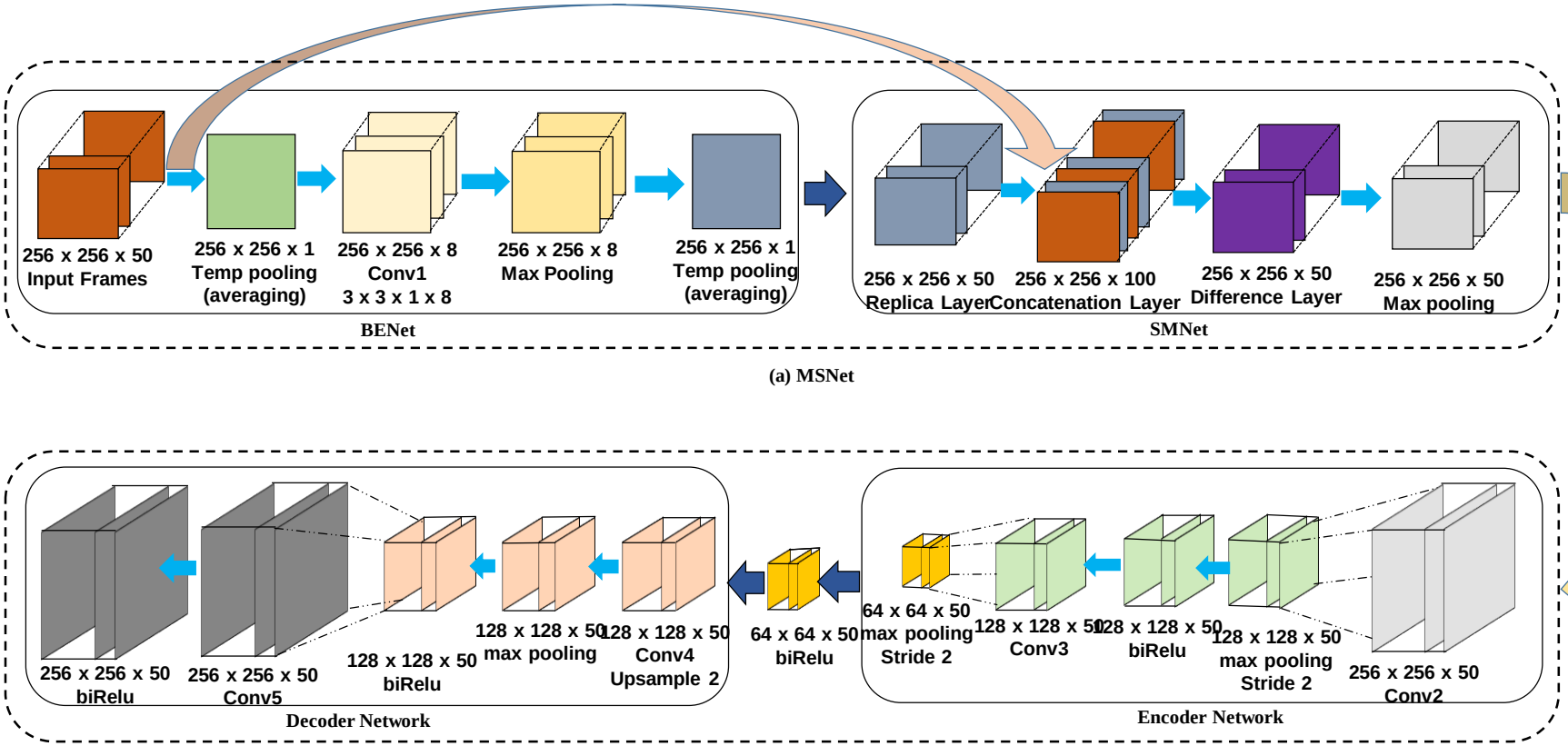


Fig. 12. Proposed network (MSFgNet) architecture for moving object detection in videos. (a) MSNet: Motion-saliency network, (b) FgNet: Foreground extraction network (*BENet: background estimation network, SMNet: saliency estimation network*).

Result Analysis

- Effectiveness (*qualitative and quantitative*) of the proposed method is validated on various challenging videos.
- Results are examined using global and cross-data training-testing approach using benchmark database.
- For global training-testing, 653 SVS (32,650 frames) are collected from CDnet-2014 and LASIESTA database.
- For cross-data training-testing, the global trained model is tested on PTIS database.

Results on CDnet-2014 database

- The CDnet-2014 dataset consists of total 53 videos with 11 different categories.
- Effectiveness of the proposed method is compared using various challenging scenario

Table III: Comparison of average {precision, recall and F-measure} with other existing methods on CDnet-2014 dataset

Methods	Avg. Precision	Avg. Recall	Avg. F-measure
DeepBS	0.8332	0.7545	0.7594
IUTIS	0.8087	0.7849	0.7821
PAWCS	0.7857	0.7718	0.7480
SuBSENCE	0.7509	0.8124	0.7453
WeSamBE	0.7955	0.7679	0.7446
FTSG	0.7657	0.7696	0.7283
SharedMod	0.8098	0.7503	0.7474
MSFgNet	**0.8502**	**0.8375**	**0.8406**

Results on CDnet-2014 database

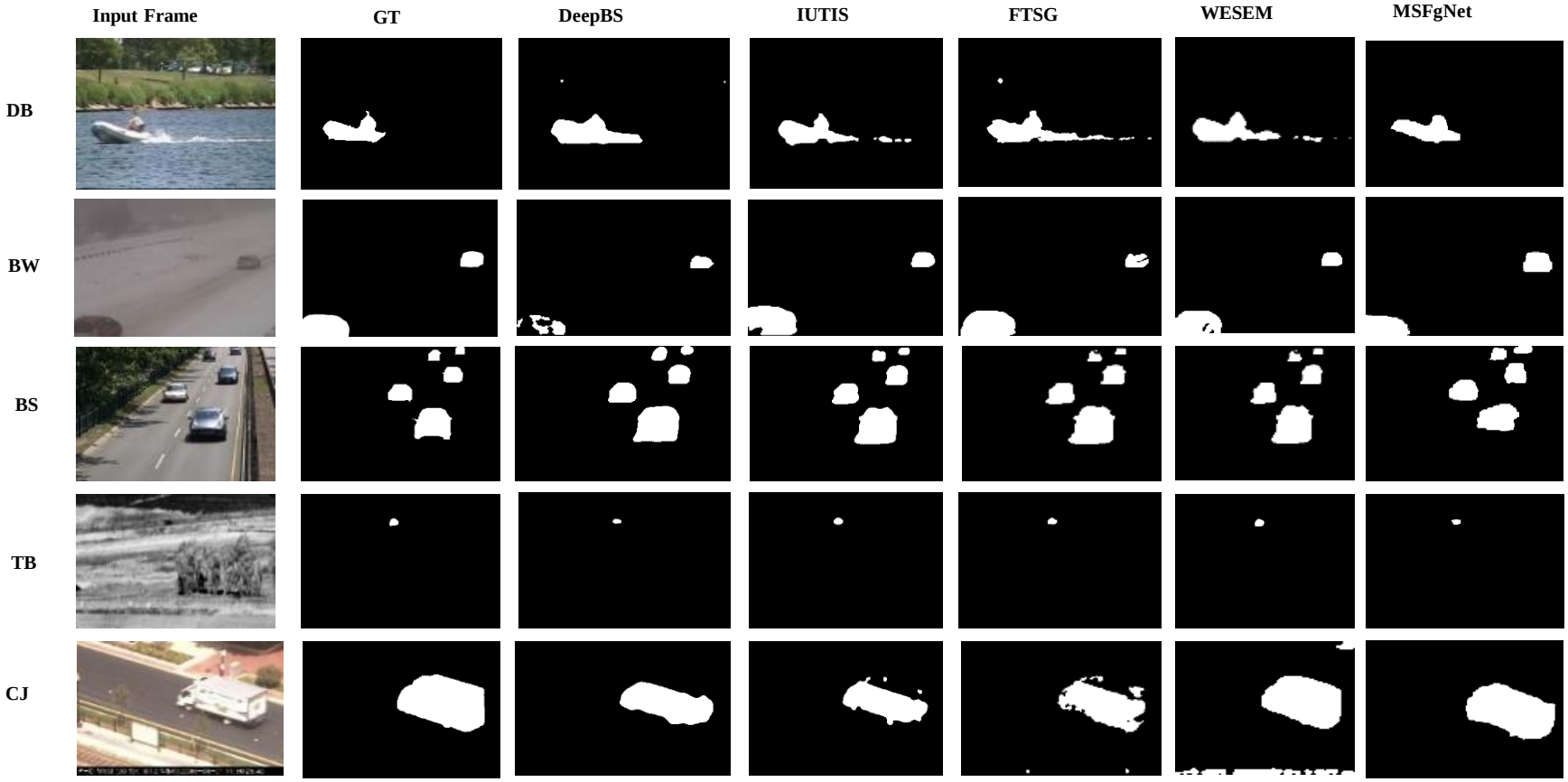


Fig. 13 : Visual comparison of various methods on CDnet-2014 dataset for foreground detection.

Results on LASIESTA database

- LASIESTA dataset consists of indoor and outdoor videos with different conditions like cloudy, rainy, snowy, etc.

Table IV: Comparison of average F-measure with other existing methods on LASIESTA dataset

Methods	Maddalena	Cuevas	Haines	Cuevas	MSFgNet
I_SI	0.74	0.78	0.88	0.88	0.92
I_CA	0.85	0.73	0.89	0.84	0.92
I_OC	0.75	0.85	0.92	0.78	0.91
I_MB	0.71	0.72	0.84	0.64	0.89
O_CL	0.87	0.86	0.82	0.87	0.88
O_RA	0.84	0.80	0.85	0.81	0.86
O_SN	0.81	0.45	0.77	0.77	0.89
O_SU	0.87	0.73	0.85	0.72	0.78
Average	**0.76**	**0.73**	**0.77**	**0.79**	**0.87**

Conclusion:

- An end-to-end CNN network trained with sub-modules of background estimation, saliency estimation and foreground segmentation network is proposed for MOD.
- The problems associated with existing methods are addressed.
- Along with qualitative and quantitative results, the computational complexity is examined.
- Performance analysis illustrates that the proposed MSFgNet is faster and outperforms the state-of-the-art methods for MOD on benchmark database.

Generative Adversarial Networks (GANs)

Ian Goofellow et al.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets,” NIPS-2014.

GANs

- **Generative:** Learn a generative model
- **Adversarial:** Trained in an adversarial setting
- **Networks:** Use Deep Neural Networks

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Why Generative Models?

- **We've only seen discriminative models so far**
  - Given an image **X**, predict a label **Y**
  - Estimates **P(Y|X)**
- **Discriminative models have several key limitations**
  - Can't model **P(X)**, i.e. the probability of seeing a certain image
  - Thus, can't sample from **P(X)**, i.e. **can't generate new images**
- **Generative models (in general) cope with all of above**
  - Can model **P(X)**
  - Can generate new images

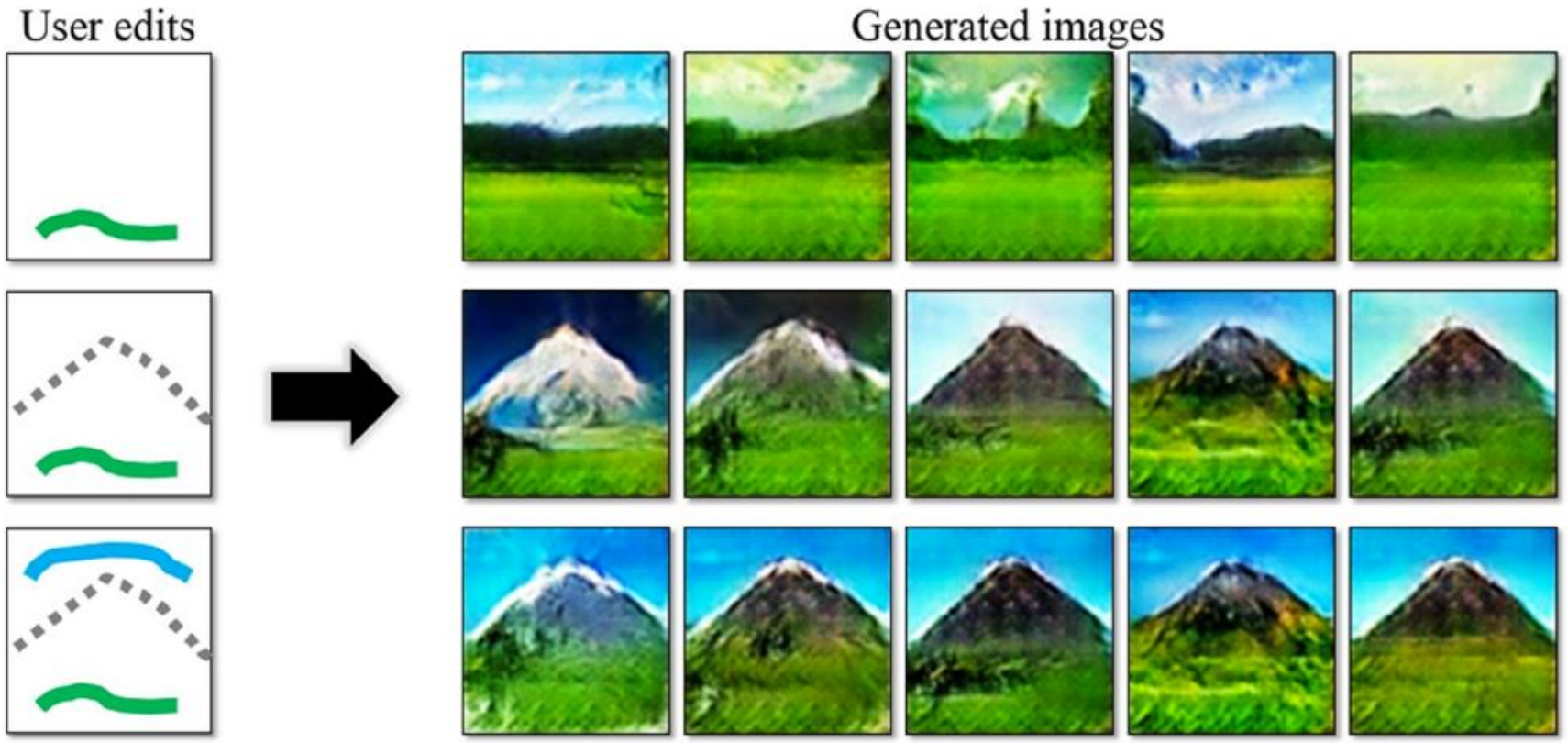
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Magic of GANs...

Which one is Computer generated?

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Magic of GANs...

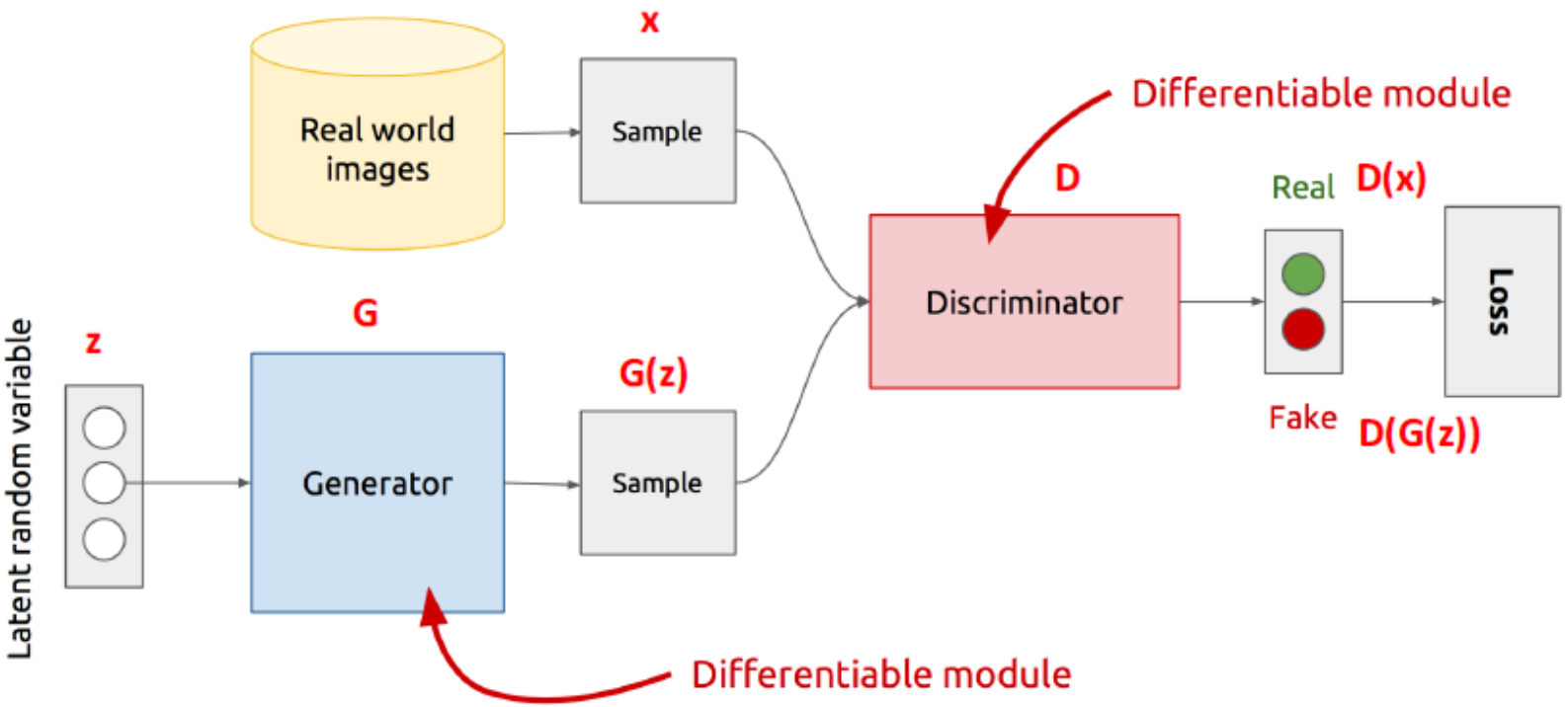


Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets,” NIPS-2014.

- **GANs extend that idea to generative models:**
  - Generator: generate fake samples, tries to fool the Discriminator
  - Discriminator: tries to distinguish between real and fake samples
  - Train them against each other
  - Repeat this and we get better Generator and Discriminator

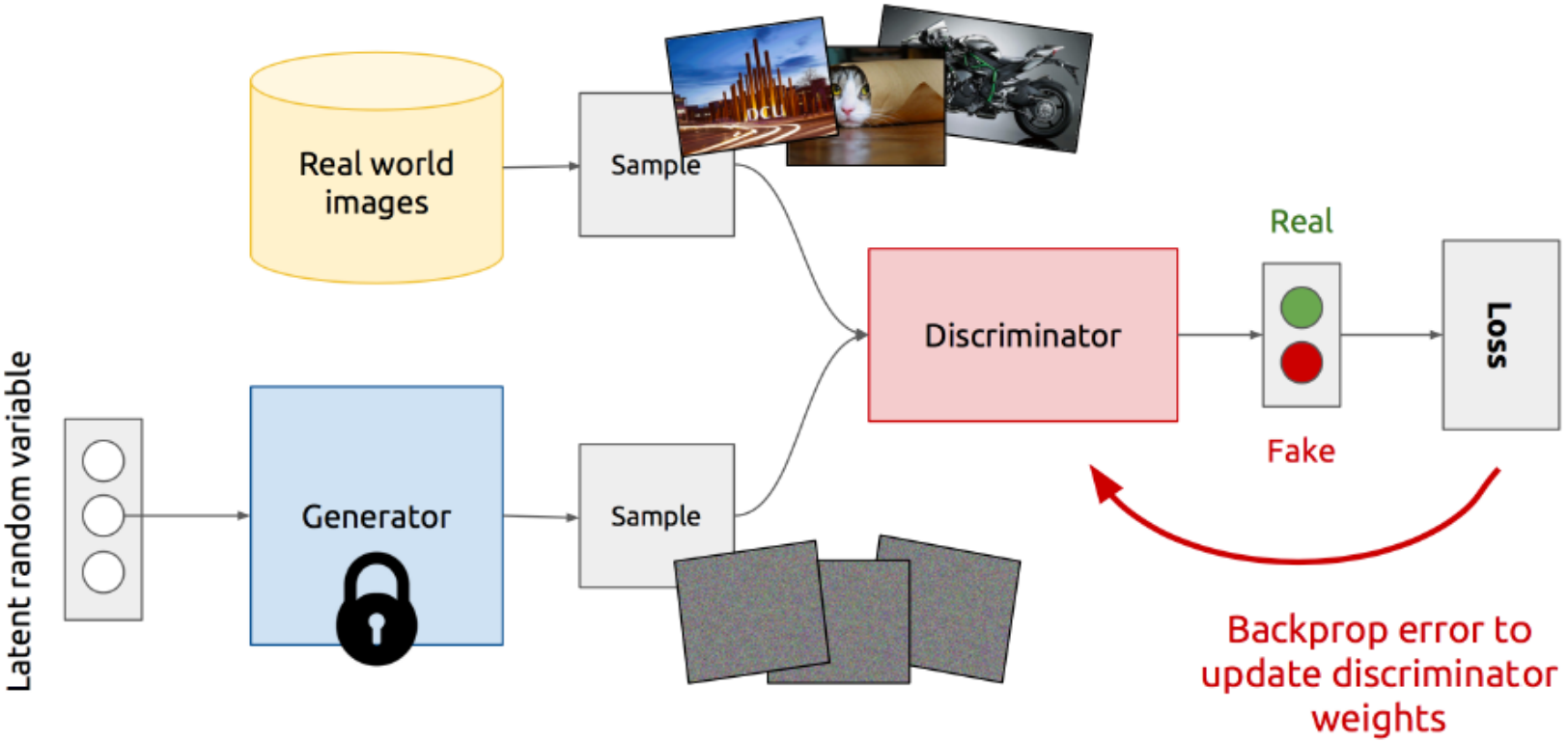
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

GAN's Architecture



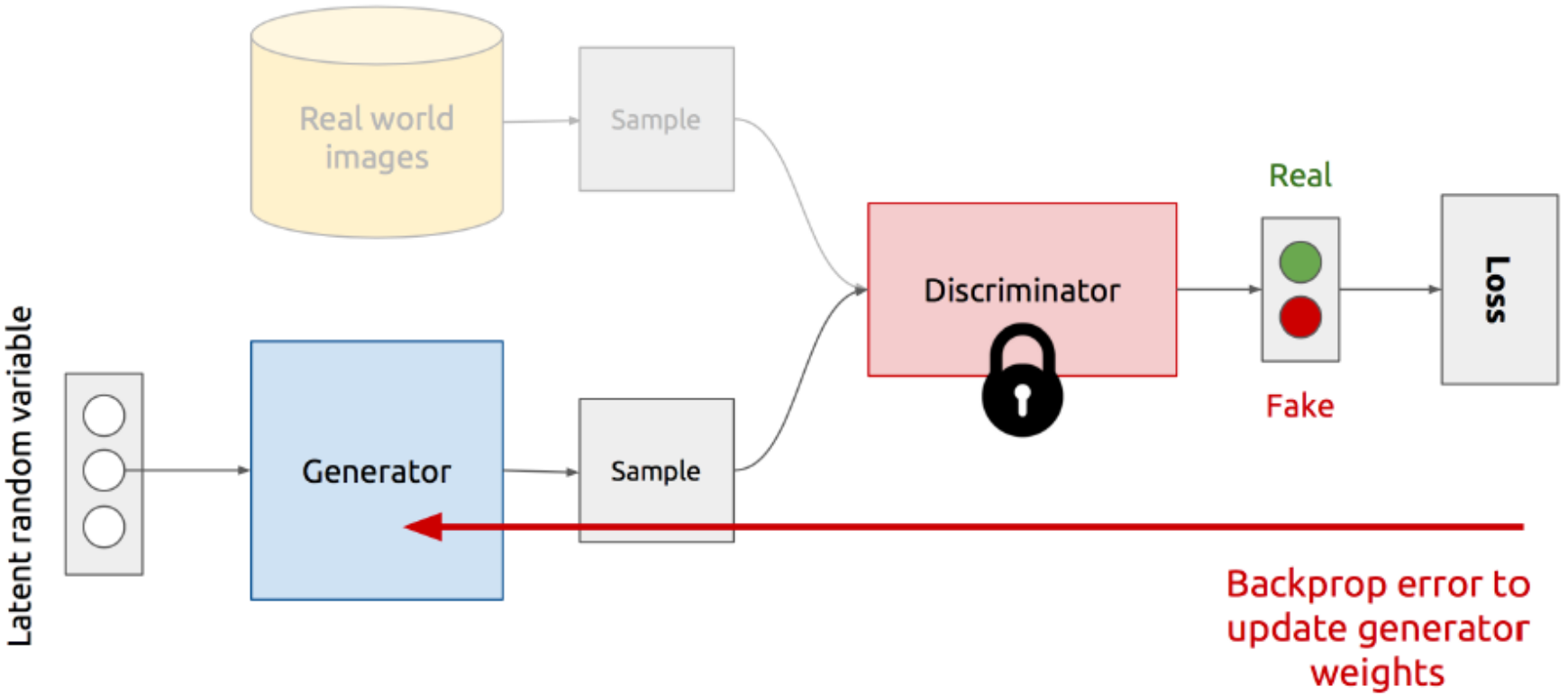
- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

Training Discriminator



Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets,” NIPS-2014.

Training Generator



Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

GAN's formulation

$$\min_G \max_D V(D,G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize its reward $\boldsymbol{V(D,G)}$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $\mathrm{D}(x) = \frac{1}{2} \;\; \forall x$

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

Discriminator updates

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

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

Generator updates

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Vanishing gradient strikes back again...

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

- $\nabla_a \log(1-\sigma(a)) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\,(1-\sigma(a))}{1-\sigma(a)} = -\sigma(a) = -D(G(z))$
- Gradient goes to 0 if D is confident, i.e. $D(G(z)) \rightarrow 0$

Minimize $-\mathbb{E}_{z\sim q(z)}[\log D(G(z))]$ for **Generator** instead (keep Discriminator as it is)

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Advantages of GANs

- Sampling (or generation) is straightforward.
- Training doesn't involve Maximum Likelihood estimation.
- Robust to Overfitting since Generator never sees the training data.
- Empirically, GANs are good at capturing the modes of the distribution.

Problems with GANs

- **Training is Hard**
  - Non-Convergence
  - Mode-Collapse

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Some real examples

Some Solutions

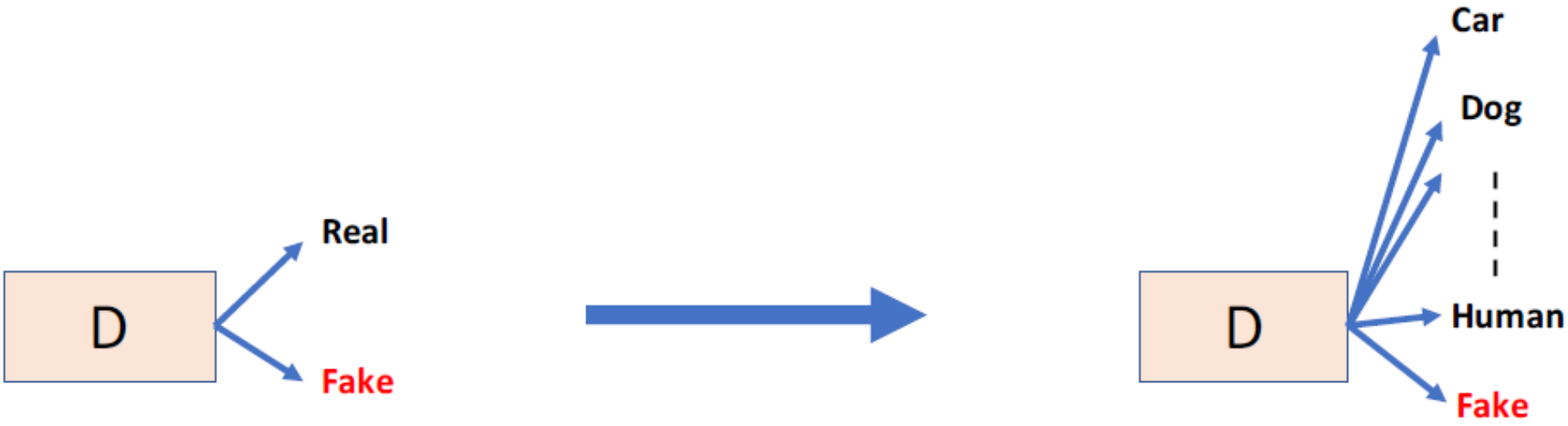
- Mini-Batch GANs
- Supervision with labels

Mini-Batch GANs

- Extract features that capture diversity in the mini-batch
  - For e.g. L2 norm of the difference between all pairs from the batch
- Feed those features to the discriminator along with the image
- Feature values will differ b/w diverse and non-diverse batches
  - Thus, Discriminator will rely on those features for classification
- This in turn,
  - Will force the Generator to match those feature values with the real data
  - Will generate diverse batches

Supervision with Labels

- Label information of the real data might help



- Empirically generates much better samples

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Alternate view of GANs

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

$$D^* = \arg\max_D V(D,G) \qquad G^* = \arg\min_G V(D,G)$$

- In this formulation, Discriminator's strategy was $D(x) \to 1, \quad D(G(z)) \to 0$

- Alternatively, we can flip the binary classification labels i.e. **Fake = 1**, **Real = 0**

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log(1 - D(x))] + \mathbb{E}_{z\sim q(z)}[\log(D(G(z)))]$$

- In this new formulation, Discriminator's strategy will be $D(x) \to 0, \quad D(G(z)) \to 1$

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Alternate view of GANs (Contd.)

- If all we want to encode is $D(x) \rightarrow 0, \quad D\big(G(z)\big) \rightarrow 1$

$$D^* = argmax_D \, \mathbb{E}_{x \sim p(x)}\big[\log\big(1 - D(x)\big)\big] + \mathbb{E}_{z \sim q(z)}\big[\log\big(D(G(z))\big)\big]$$

We can use this

$$D^* = argmin_D \, \mathbb{E}_{x \sim p(x)} \log\big(D(x)\big) + \mathbb{E}_{z \sim q(z)}\Big[\log\Big(1 - D\big(G(z)\big)\Big)\Big]$$

- Now, we can replace cross-entropy with any loss function **(Hinge Loss)**

$$D^* = argmin_D \, \mathbb{E}_{x \sim p(x)} D(x) + \mathbb{E}_{z \sim q(z)} \max\Big(0, m - D\big(G(z)\big)\Big)$$

- And thus, instead of outputting probabilities, Discriminator just has to output :-
  - High values for fake samples
  - Low values for real samples

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets,” NIPS-2014.

- **Applications**
  - Image-to-Image Translation
  - Text-to-Image Synthesis
  - Face Aging

Image-to-Image Translation

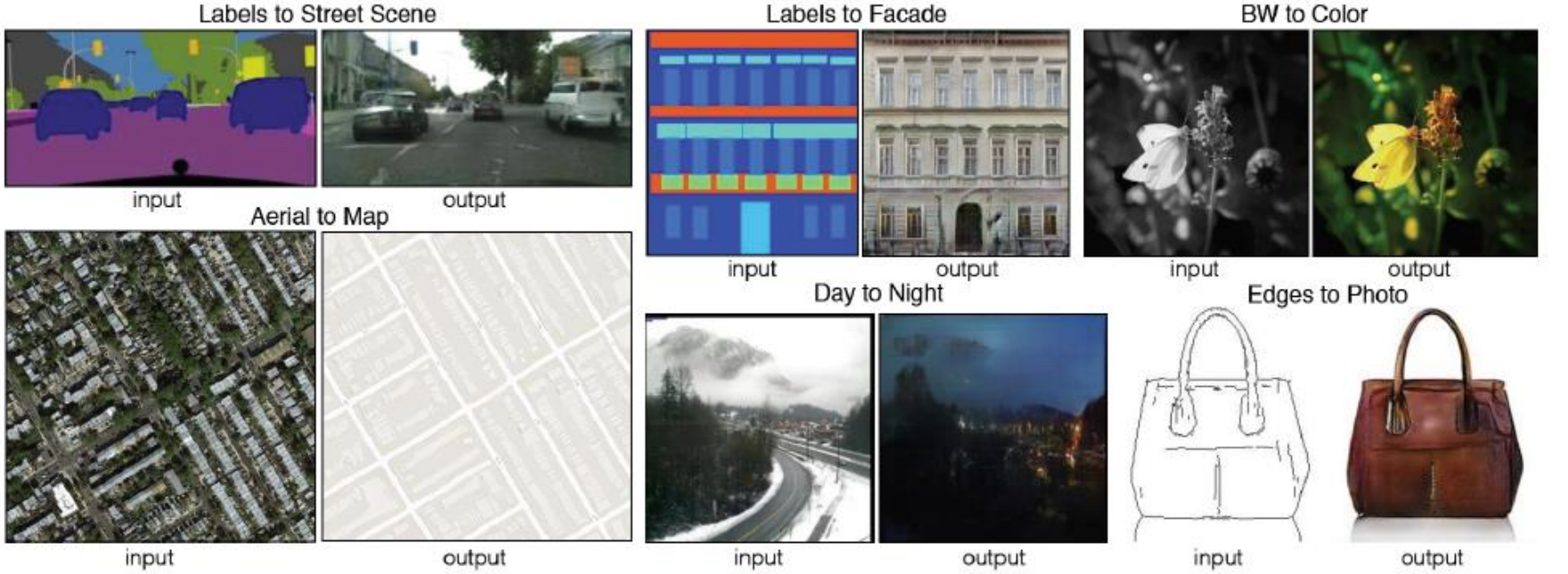


Figure 1 in the original paper.

Face Aging with Conditional GANs

- Differentiating Feature: Uses an *Identity Preservation Optimization* using an auxiliary network to get a better approximation of the latent code (z*) for an input image.
- Latent code is then conditioned on a discrete (one-hot) embedding of age categories.

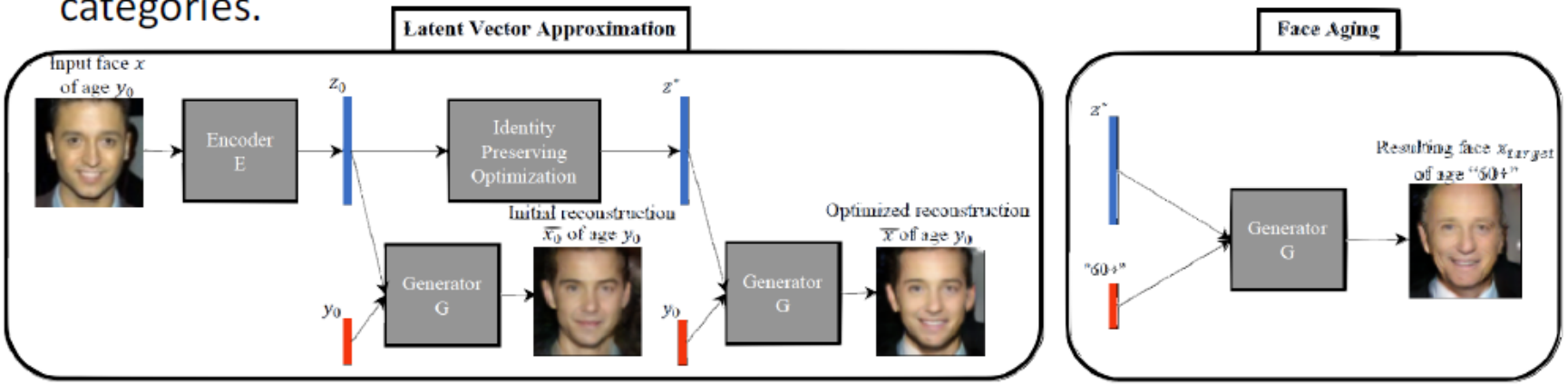


Figure 1 in the original paper.

Hazy Image

Proposed Method
S = 0.63, P = 17.88, C = 12.00

DehazeNet (TIP 2016)
S = 0.23, P = 12.41, C = 20.64

MSCNN (ECCV 2016)
S = 0.31, P = 13.60, C = 17.84

CAP (TIP 2014)
S = 0.24, P = 12.77, C = 19.87

DChP (TPAMI 2011)
S = 0.34, P = 14.55, C = 17.54

RI-GAN: An End-to-End Network for Single Image Haze Removal

NTIRE Image De-hazing Competition

#270 Labs Participanted (Including MIT Lincoln Laboratory)

#23 Labs were in final round

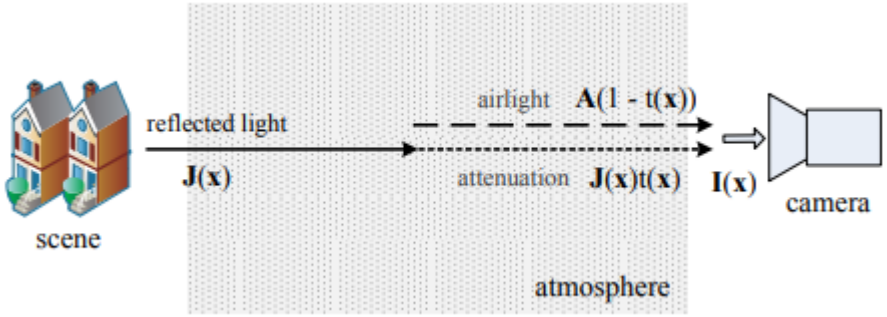
We secured **10th Rank Internationally (MIT Lincoln Lab got 11th Rank)**

1st Rank within the INDIA

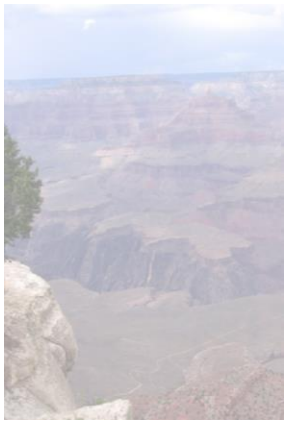
Publication:

1. Akshay Dudhane, Harshjeet Singh Aulakh and Subrahmanyam Murala. “RI-GAN: An End-to-End Network for Single Image Haze Removal”. In CVPR-W 2019.

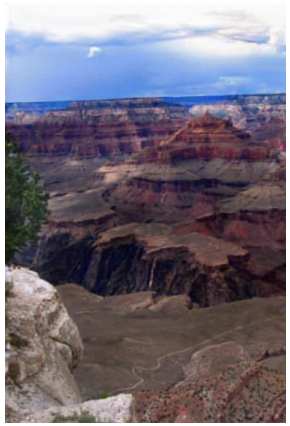
Atmospheric Scattering Model



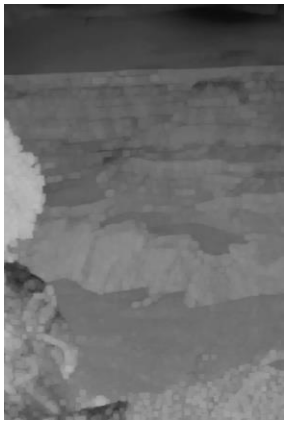
$$\mathbf{I}(\mathbf{x}) = \mathrm{J}(\mathbf{x})t(\mathbf{x}) + \mathbf{A}\big(1 - t(\mathbf{x})\big)$$



I



J



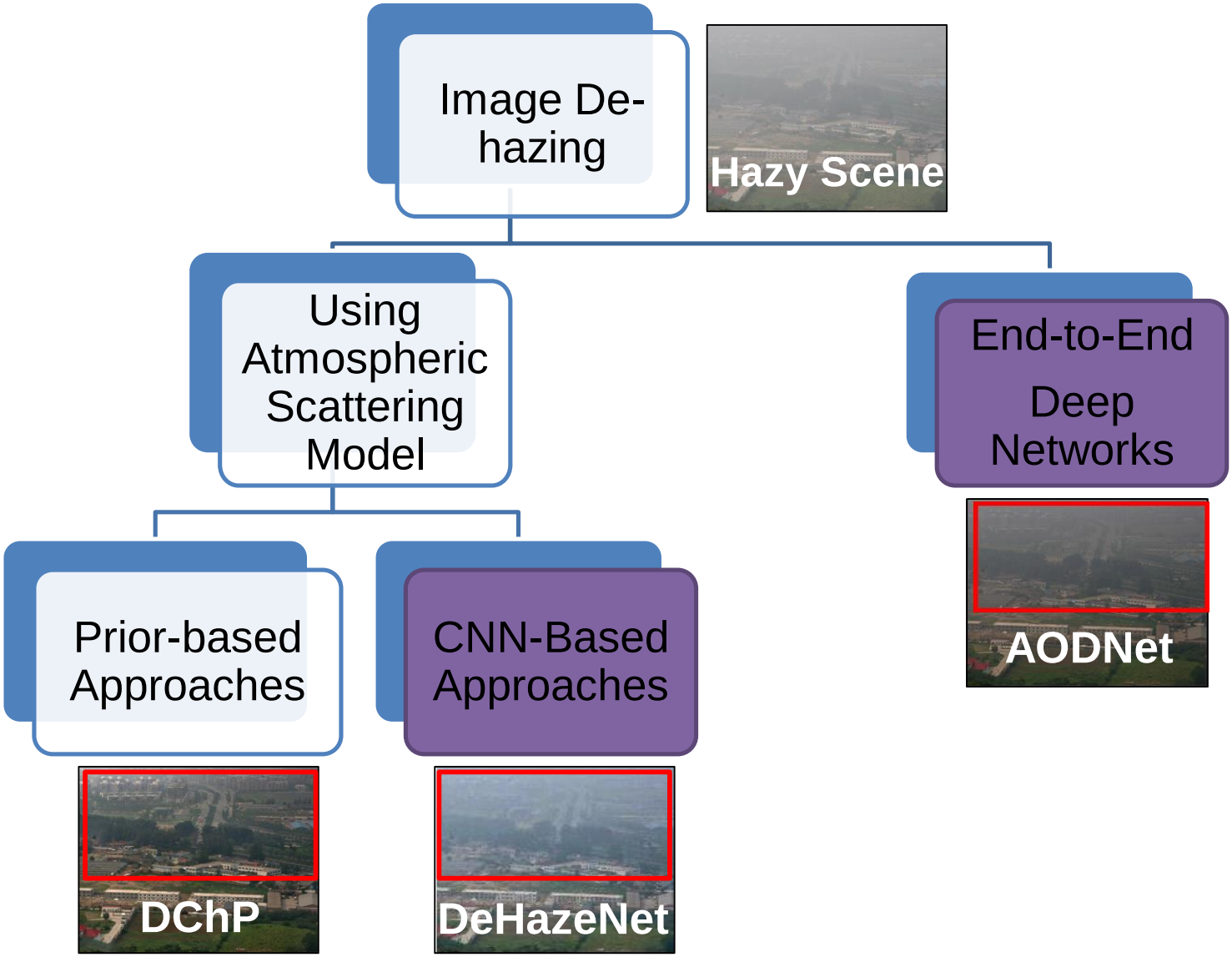
t

x = Image pixel location, **I =** is a observed hazy scene.

J = Actual haze-free scene, t = Scene transmission map. $\mathbf{J}(\boldsymbol{x})\boldsymbol{t}(\boldsymbol{x})$ = is called direct attenuation.

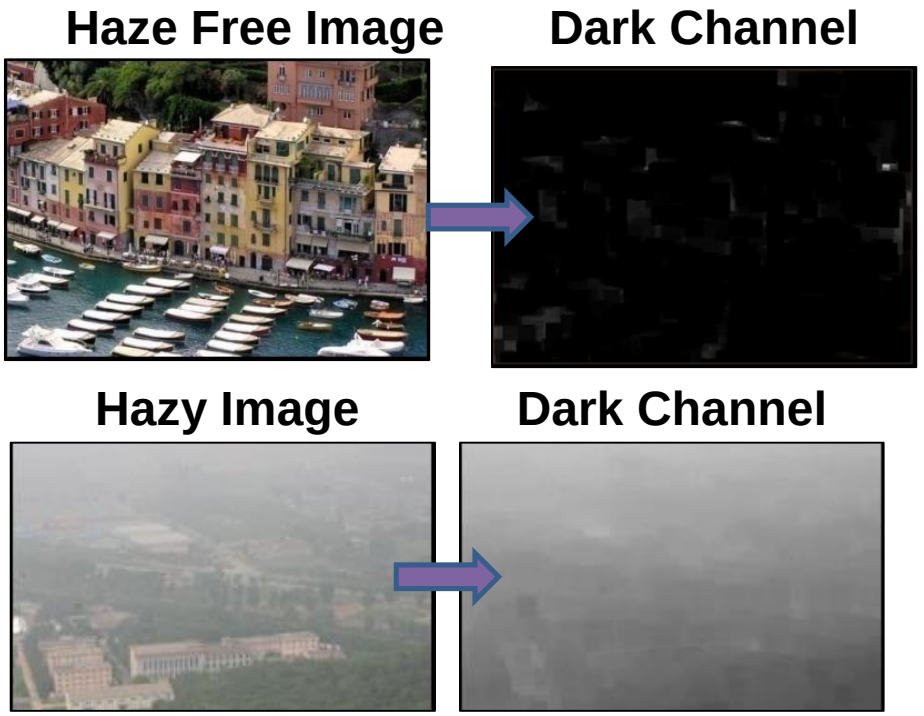
A = Scene Atmospheric Light.

Literature Survey



- Haze relevant hand-crafted priors
  - He *et al.,* "Dark channel prior", *CVPR 2008.*
  - Haung *et al.,* "Visibility Restoration of Single Hazy Images", *T-CSVT 2014.*
  - Zhu *et al.*, "A fast single image haze removal algorithm using color attenuation prior", *IEEE TIP, 2015.*
  - Tang *et al.,* "Investigating Haze-relevant Features in A Learning Framework for Image Dehazing", *CVPR 2014.*
- Deep learning-based approaches:
  - Cai *et al.,* "DehazeNet: An End-to-End System for Single Image Haze Removal", *TIP 2016*
  - Ren *et al.*, "Single Image Dehazing via Multi-Scale Convolutional Neural Networks", *ECCV 2016.*
- End-to-end deep learning-based approaches:
  - B. Li et al., "Aod-net: All-in-one dehazing network", *ICCV 2017*

Dark Channel Prior



Observation:

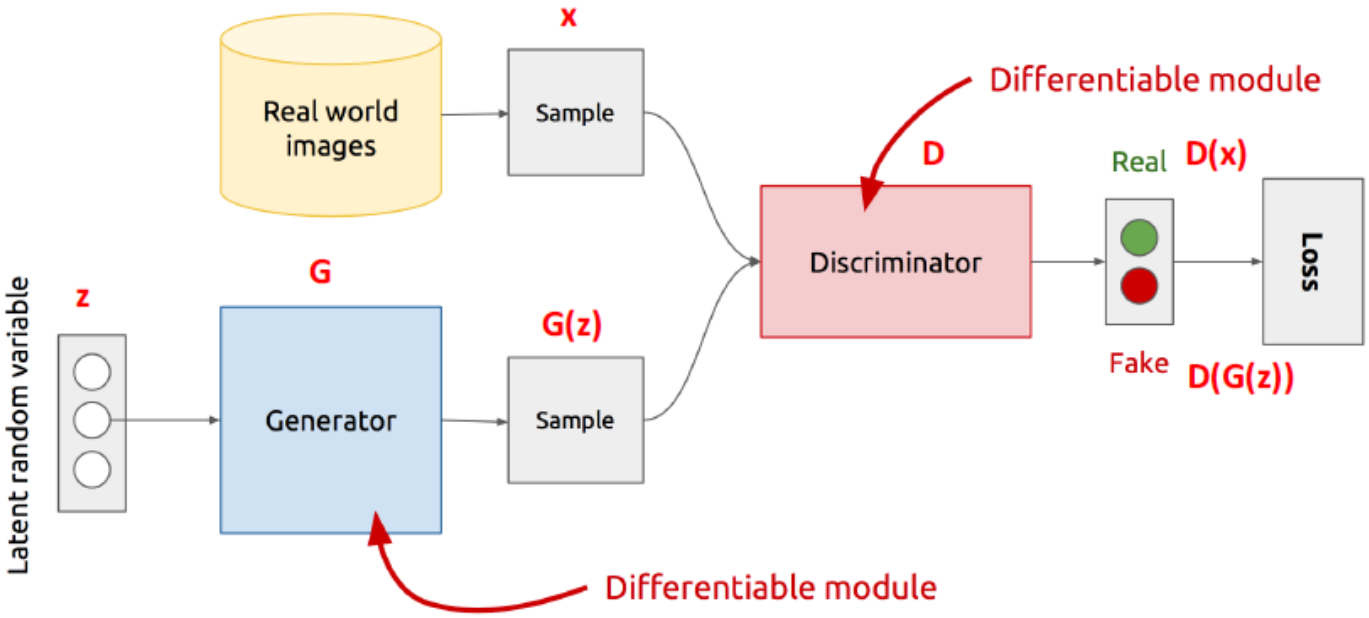
- In most of the non sky patches, at least one color channel has some pixels whose intensity are very low and close to zero.
- Equivalently, the minimum intensity in such a patch is close to zero.

K. He et al.. "**Single Image Haze Removal using Dark Channel Prior**". IEEE TPAMI 2011.

- **Assumption:**
  - Atmospheric model can be modeled in end-to-end deep network.
  - Availability of dense paired training data.

- **Contribution:**
  - End-to-end conditional generative adversarial network named as RI-GAN is proposed for image de-hazing.
  - A novel generator network is proposed which is designed using a combination of both residual and inception module.
  - A novel discriminator network is designed using dense connections in the residual block.
  - A combination of structural similarity index (SSIM) loss and edge loss are incorporated along with the L1 loss to optimize the network parameters.

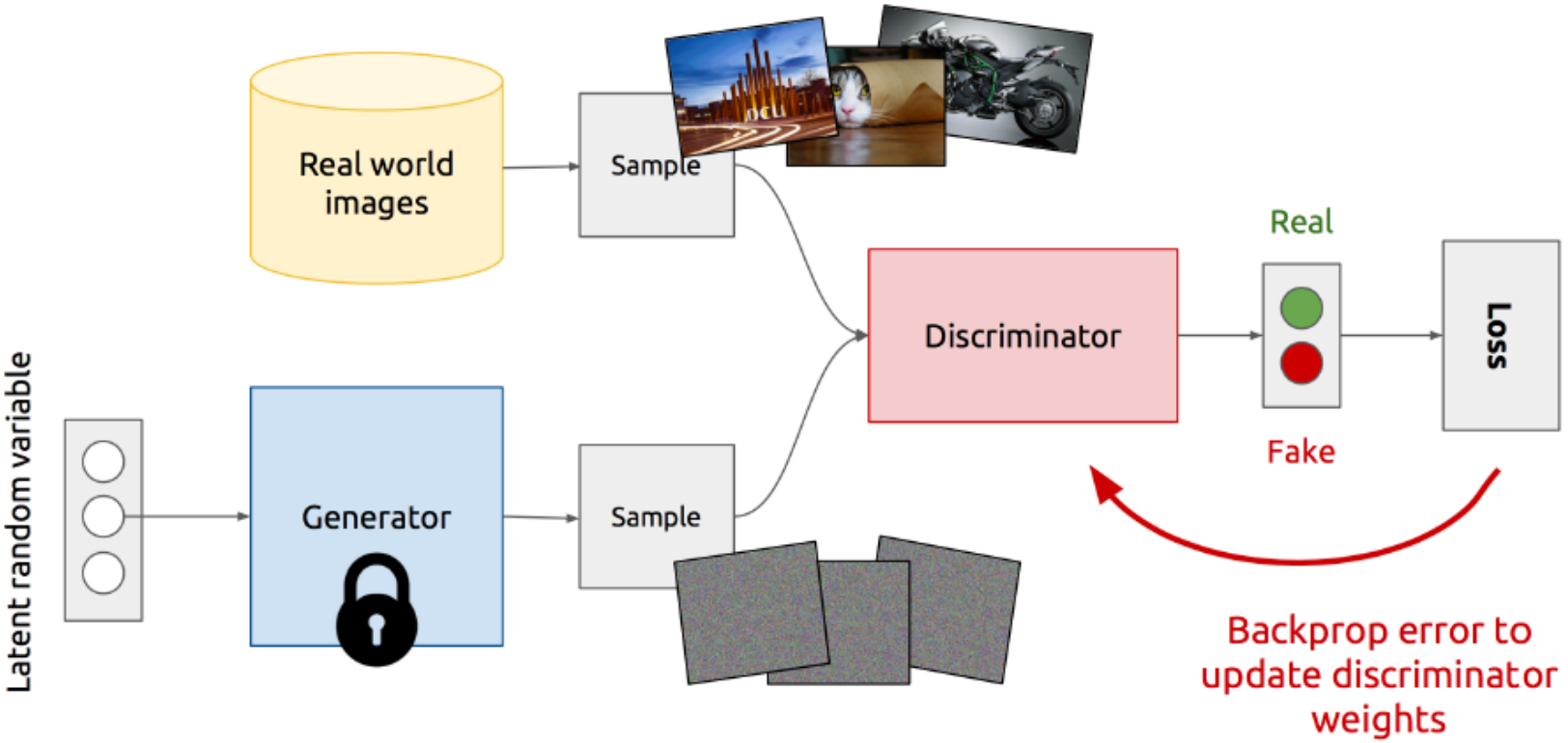
GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform).
- **Z** can be thought as the latent representation of the image.

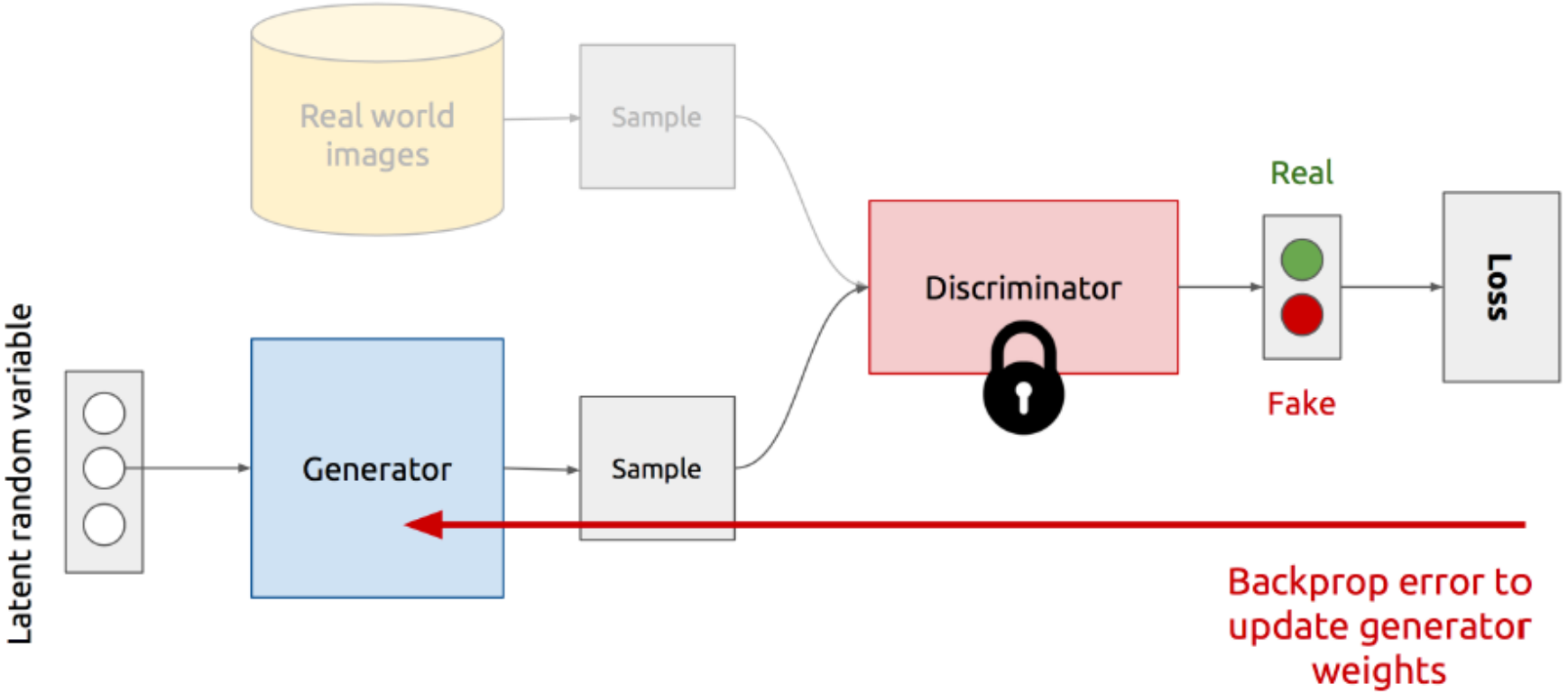
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Training Discriminator



Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets,” NIPS-2014.

Training Generator



Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets,” NIPS-2014.

Proposed Approach with GAN

Discriminator Loss Updating (Stochastic Gradient Ascent)

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

Generator Loss Updating (Stochastic Gradient Descent)

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

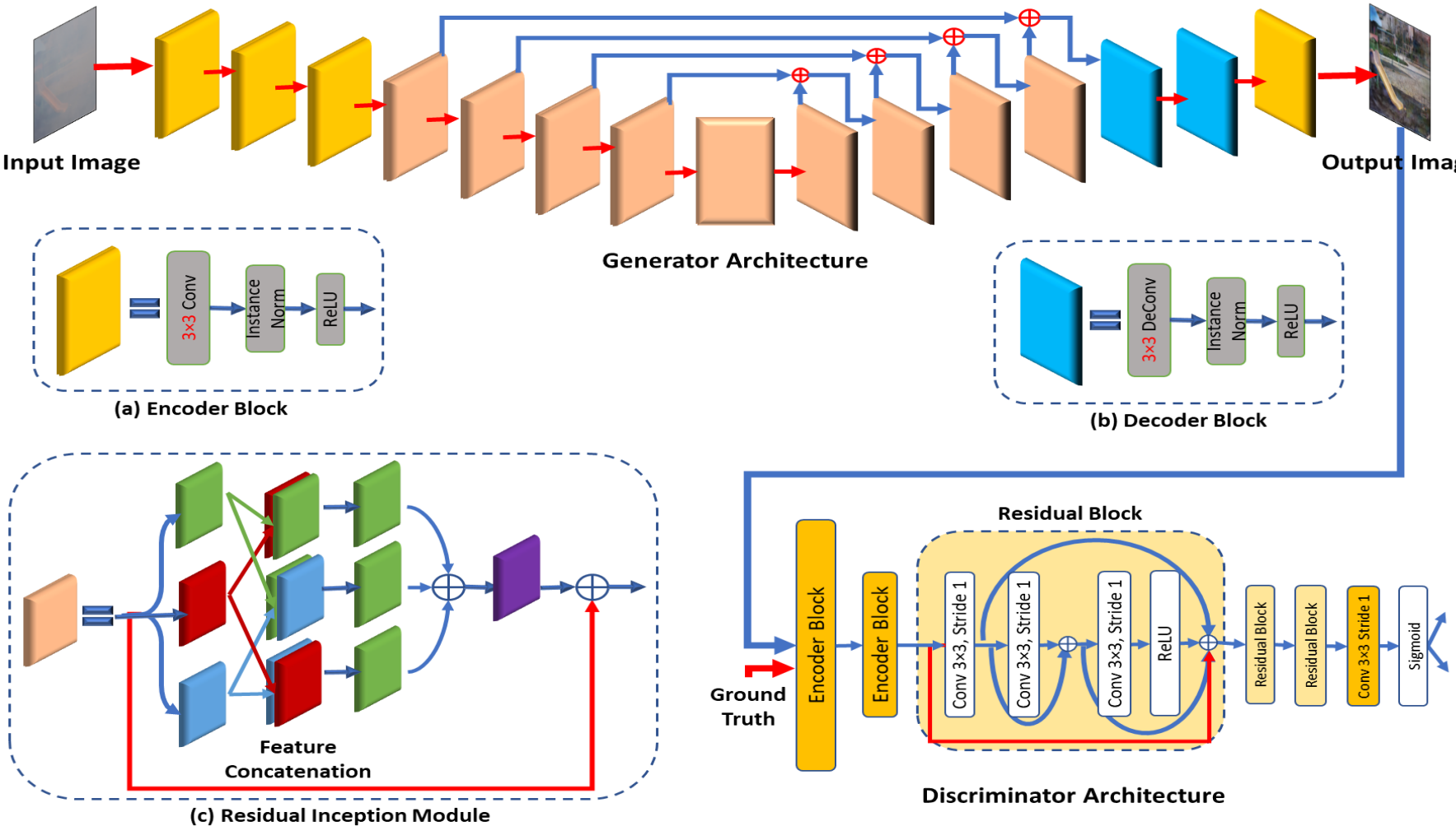
$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)} \left[\log\left(1 - D\left(G(z)\right)\right) \right]$$

- $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- Gradient goes to 0 if D is confident, i.e. $D\left(G(z)\right) \rightarrow 0$

Minimize $-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]$ for **Generator** instead (keep Discriminator as it is)

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, "Generative Adversarial Nets," NIPS-2014.

Proposed Network



Proposed Loss Functions

- **SSIM Loss**

$$\ell_{SSIM}(G) = 1 - SSIM(G(x), y)$$

$$l(G(x), y) = \frac{2\mu_{G(x)}\mu_y + C_1}{\mu_{G(x)}^2 + \mu_y^2 + C_1}$$

$$c(G(x), y) = \frac{2\sigma_{G(x)}\sigma_y + C_2}{\sigma_{G(x)}^2 + \sigma_y^2 + C_2}$$

$$s(G(x), y) = \frac{\sigma_{G(x)y} + C_3}{\sigma_{G(x)}\sigma_y + C_3}$$

$$SSIM = \frac{\left(2\mu_{G(x)}\mu_y\right)\left(2\sigma_{G(x)y} + C_2\right)}{\left(\mu_{G(x)}^2 + \mu_y^2 + C_1\right)\left(\sigma_{G(x)}^2 + \sigma_y^2 + C_2\right)}$$

- **Edge Loss**

$$\ell_{Edge}(G) = \| E_{G(x)} - E_y \|_1$$

Proposed Loss Functions

- **Over all loss function**

$$\mathcal{L}(G, D) = l_{cGAN}(G, D) + \lambda \cdot l_{SSIM}(G) + l_{Edge}(G) + \lambda \cdot l_{L1}(G)$$

$$G^* = \arg \min_G \max_D \mathcal{L}(G, D)$$

Training of the Proposed RI-GAN

- The outdoor NTIRE2018 dehazing challenge (35) [5] and NTIRE2019 dehazing challenge (45) [4].
- Combinedly, 100 synthetic hazy images of NYU depth database and 80 NTIRE database are used to train the proposed RI-GAN.
- Proposed network is trained for 200 epochs on a computer having 4.20 GHz Intel Core i7 processor and NVIDIA GTX 1080 8GB GPU.

Experimental Results

Team	User (+entry)	PSNR	SSIM
iPAL-AtJ	moonriverLucy	**20.258**	**0.657**
iPAL-COLOR	DH-IRCNN_123_CEDH	19.923	0.653
MT.MaxClear	ucenter52	19.469	0.652
BMIPL-UNIST-DW-1	Sprite+Ours	18.842	0.633
xddqm	Untitled Folder	18.521	0.640
ECNU	emmm+dpn_best	17.826	0.617
MOMOCV	meshpop	17.177	0.564
BMIPL-UNIST-DW-2	BMIPL-PDW+Hazing	16.857	0.610
BOE-IOT-AIBD	BOE-IOT-AIBD	16.780	0.612
MAHA@IIT	akshay.aad16	16.472	0.548
FastNet	tzofi+submission	16.371	0.569
IVL1	IVL+submission16	16.194	0.601
ecsuiplab1	san_santra+up_4	16.152	0.564
IPCV IITM	maitreya_ipcv+final	16.126	0.595
shh	sunhee+res	16.055	0.562
IVL2	IVL+submission17	15.801	0.600
ecsuiplab2	ranjanisi+ranjan	15.969	0.539
Alex_SDU	wang_cheng	15.936	0.557
hcilab	hcilab+final	15.122	0.580
IMag	dxllx+t88	14.928	0.555
XZSYS	ChuanshengWang	14.338	0.491
Vintage	jptarel+simple1	14.021	0.529

Table 1. NTIRE 2019 Challenge dehazing results and final rankings on DENSE-HAZE test data.

Experimental Results

Hazy Image

Proposed Method

DChP [1] (TPAMI 2011)

C^{2}MSNet [2] (WACV-18)

GFN [3] (CVPR 2018)

AODNet [4] (ICCV 2017)

[1] K. He, J. Sun, and X. Tang, "**Single image haze removal using dark channel prior**," IEEE TPAMI, 2011.
[2] Akshay Dudhane and Subrahmanyam Murala. "**C2MSNet: A Novel Approach for Single Image Haze Removal**". In IEEE WACV 2018.
[3] W. Ren et al.,"**Gated fusion network for single image dehazing**," in CVPR 2018.
[4] B. Li, X. et al., "**Aod-net: All-in-one dehazing network**," in ICCV 2017

Experimental Results

Hazy Image

Proposed Method

DChP [1] (TPAMI 2011)

C²MSNet [2] (WACV-18)

GFN [3] (CVPR 2018)

AODNet [4] (ICCV 2017)

[1] K. He, J. Sun, and X. Tang, "**Single image haze removal using dark channel prior**," IEEE TPAMI, 2011.
[2] Akshay Dudhane and Subrahmanyam Murala. "**C2MSNet: A Novel Approach for Single Image Haze Removal**". In IEEE WACV 2018.
[3] W. Ren et al.,"**Gated fusion network for single image dehazing**," in CVPR 2018.
[4] B. Li, X. et al., "**Aod-net: All-in-one dehazing network**," in ICCV 2017

Experimental Results

Hazy Image

Proposed Method

CAP [2]
(TIP 2014)

C^{2}MSNet [3]
(WACV 2018)

MSCNN [8]
(ECCV 2016)

Experimental Results

Hazy Image

Proposed Method

C2MSNet [1] (WACV 2018)

AODNet [2] (ICCV 2017)

GFN [3] (CVPR 2018)

CycleDehaze [4] (CVPR 2018)

Conclusion

1. In this work, we propose an end-to-end GAN for single image haze removal.

2. A novel generator network is designed using residual and inception principles named as Residual Inception GAN (RI-GAN).

3. Also, a novel discriminator network using dense residual module is proposed to discriminate between fake and real samples.

4. We propose a combination of SSIM and edge loss while training the proposed RI-GAN.

5. Performance of the proposed RI-GAN has been evaluated on NTIRE2019 challenge dataset [4], D-Hazy [1], SOTS [22] and real-world hazy scenes.

6. The qualitative analysis has been carried out by analyzing and comparing the results of proposed RI-GAN with existing state-of-the-art methods for image de-hazing.

7. Experimental analysis shows that the proposed RI-GAN outperforms the other existing methods for image de-hazing.

CDNet: Single Image De-hazing using Unpaired Adversarial Training

Akshay Dudhane and Subrahmanyam Murala
Computer Vision and Pattern Recognition Lab,
IIT Ropar, India

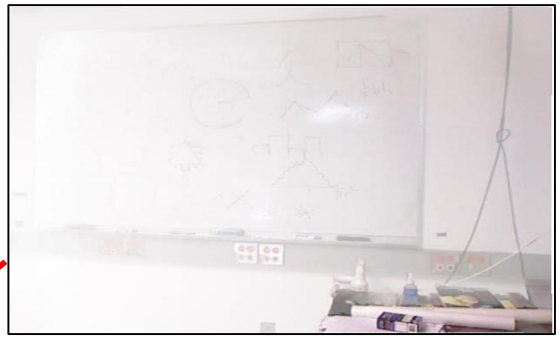
Problems identified in existing approaches

Lack of natural hazy image training set

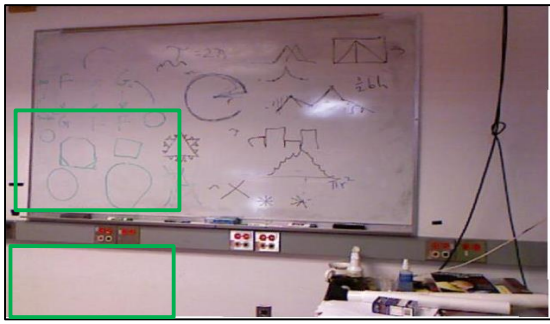
Failure in estimation of accurate Transmission map

Color distortion in recovered haze-free images.

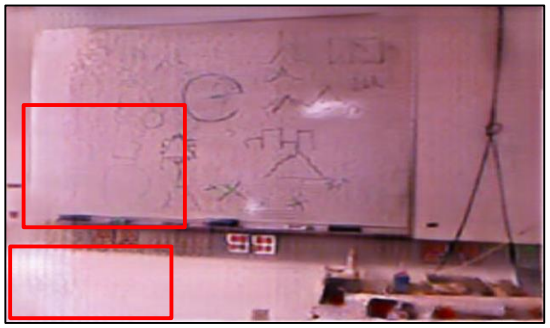
Hazy image



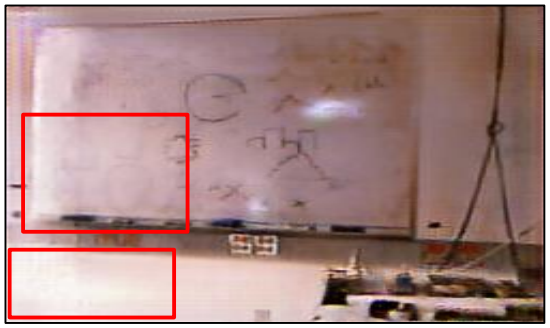
Ground truth haze-free image



CycleGAN [7]



CycleDehaze [8]



[7] J.-Y. Zhu *et al.* "Unpaired image-to-image translation using cycle-consistent adversarial networks". In ICCV 2017.
[8] D. Engin *et al.* "Cycle-dehaze: Enhanced CycleGAN for single image dehazing. In CVPR workshops 2018.

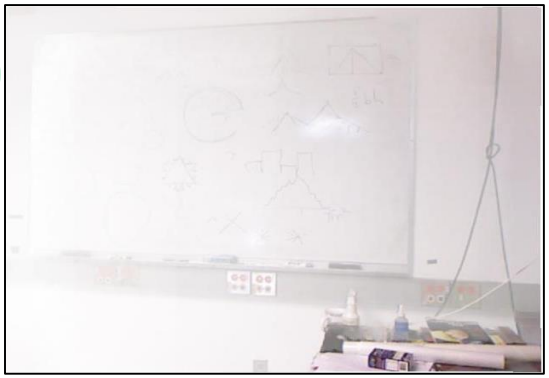
Proposed Solution

Object-level transmission map

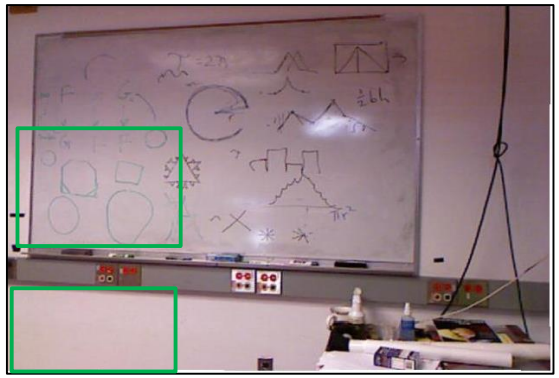
Unpaired training using random set of natural hazy and haze-free images

Distortion less haze-free image

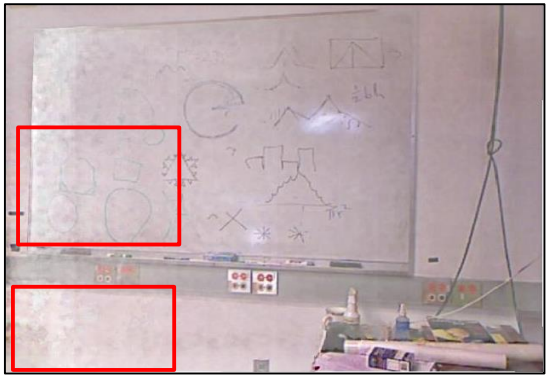
Hazy image



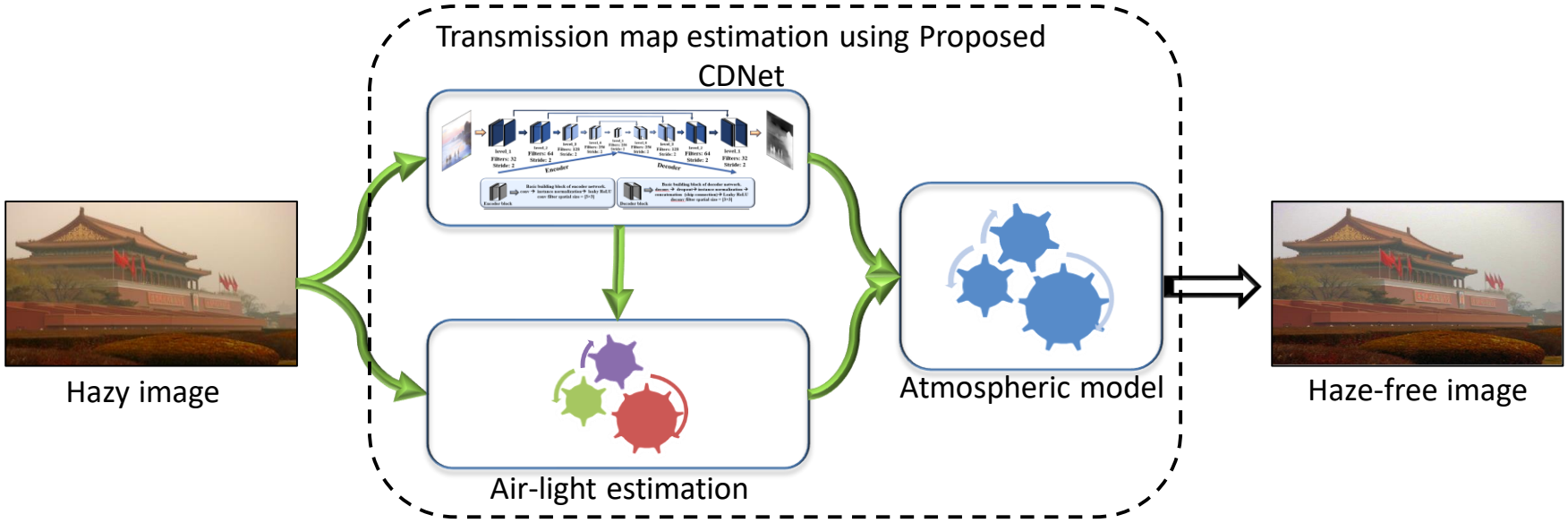
Ground truth haze-free image



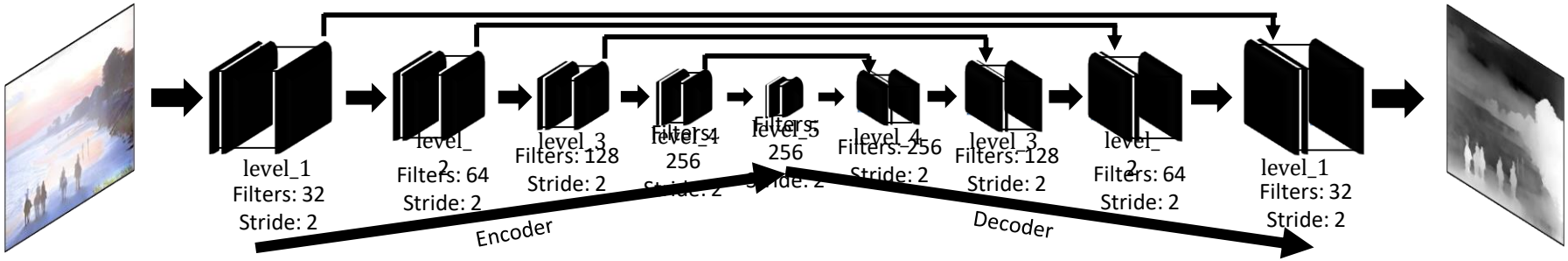
Proposed method



Overview of the Proposed Method



Proposed CDNet for transmission map estimation



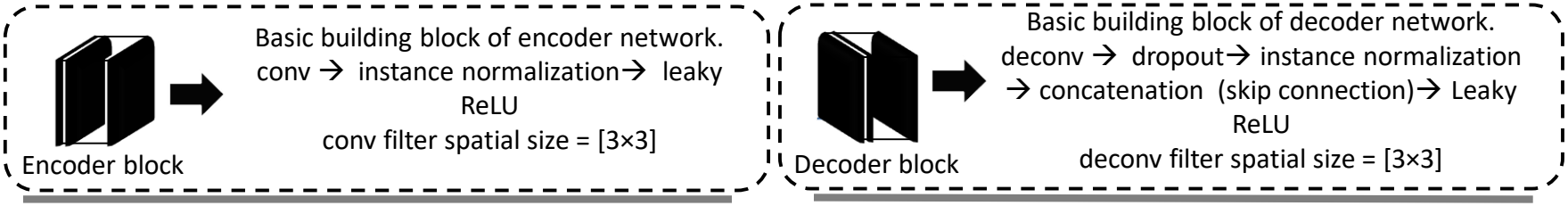
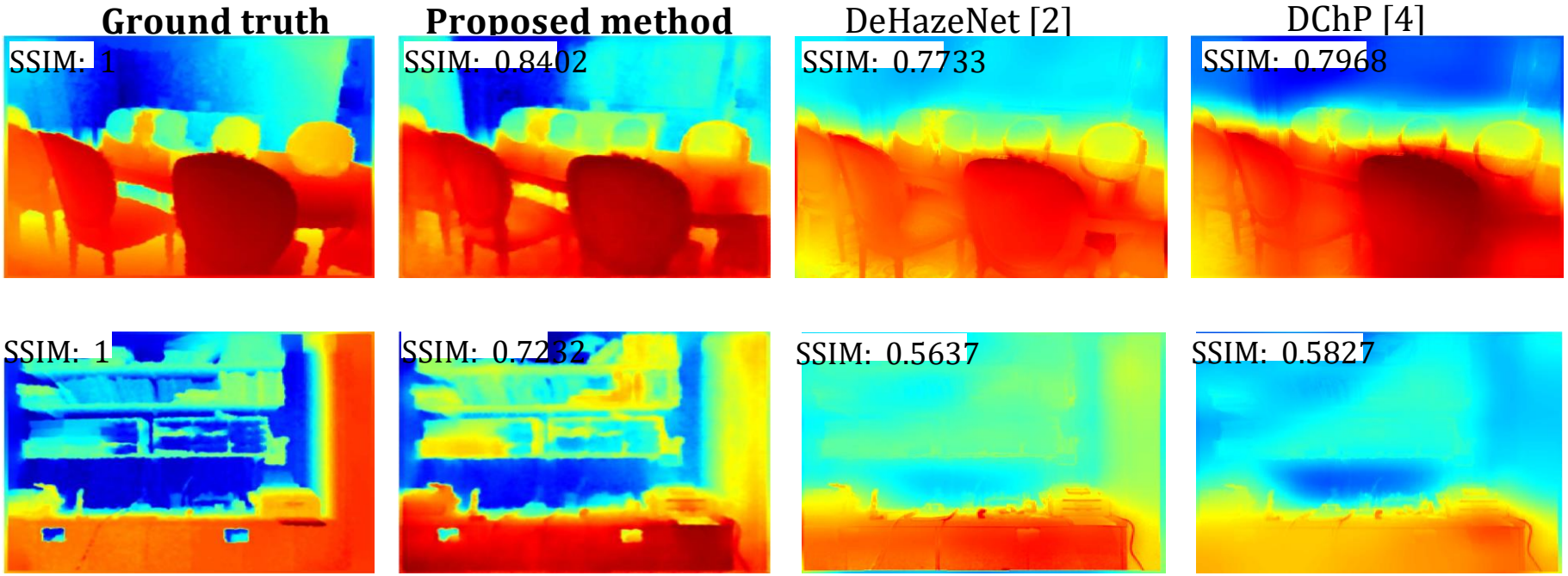


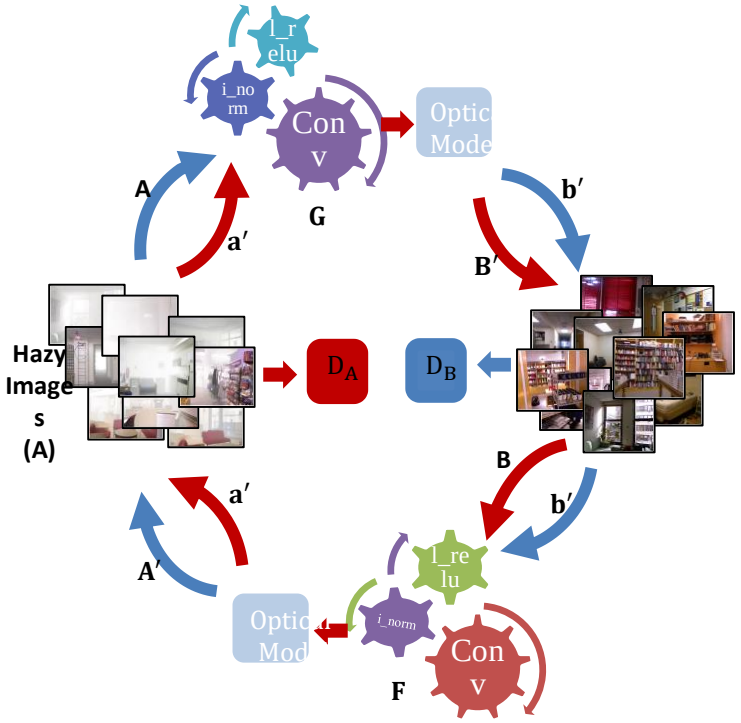
Fig. 1: Architecture of generator network used in proposed CDNet

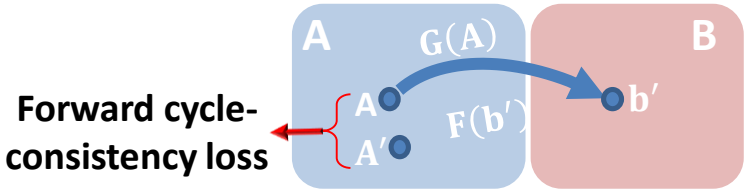
Effect of proposed CDNet

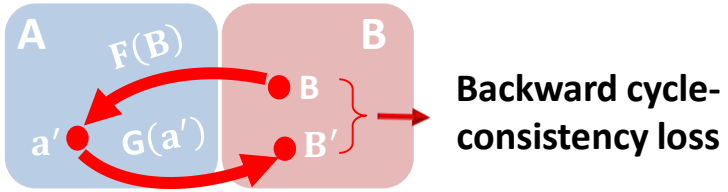


[2] B. Cai *et al.*, "Dehazenet: An end-to-end system for single image haze removal". IEEE Transactions on Image Processing, 2016.
[4] K. He *et al.*, "Single image haze removal using dark channel prior". IEEE T-PAMI, 2011.

Unpaired training for single image haze removal







Cyclic losses

i_norm: Instance Normalization layer,
Conv: Convolution layer,
l_relu: Leacky ReLU,
G and F: Generator networks
D_A and D_B : Discriminator networks

Fig. 2: Unpaired training approach for the proposed CDNet. G and F depict the A→B and B→A generators respectively. DA and DB represents A→B and B→A discriminators respectively. A: Hazy image and B: Haze free image. *Note: A blue colored arrow indicates hazy to clean image cycle and a red colored arrow indicates clean to hazy image cycle.*

Database Information

D-Hazy Database [9]

- Dense haze
- 1,449 hazy and haze-free images.
- 1,449 respective simulated depth maps.

Synthetic outdoor hazy images (SOHI) [11]

- Dense haze
- 4,035 hazy and haze-free images.

SOTS Database [10]

- Dense haze
- 4,035 hazy and haze-free images.

Set of real-world hazy images [12]

- 15 real-world hazy images for testing.

[9] C. Ancuti, *et al.,* D-hazy: A dataset to evaluate quantitatively dehazing algorithms. In ICIP, 2016.
[10] B. Li *et al.,* Reside: A benchmark for single image dehazing. arXiv preprint arXiv:1712.04143, 2017.
[11] J. Deng *et al.,* Imagenet: A large-scale hierarchical image database. In CVPR 2009.
[12] https://www.google.com/

Training details

Network training details

- Un-paired 1,000 real-world hazy and haze-free images.
- 200 hazy and haze-free images from D-Hazy database (shuffled).
- Learning rate 0.0001
- Epochs 200

Network testing details

- 1249 images from D-Hazy images
- 500 images from SOTS database
- 1000 images from SOHI database
- 15 real-world hazy images

Machine details

- NVIDIA DGX Station
- processor 2.2 GHz, Intel Xeon E5-2698 (20-Core)
- NVIDIA Tesla V100 416 GB GPU

Quantitative analysis

Table 1: Quantitative analysis of single image haze removal on D-hazy and SOTS database.

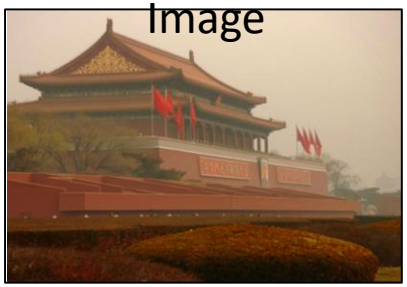
Method	D-Hazy (1249 images)			SOTS (500 images)		
	SSIM	PSNR	CIEDE2000	SSIM	PSNR	CIEDE2000
C2MSNet [1]	0.7201	12.7147	14.3162	0.8152	20.1186	8.3060
DehazeNet [2]	0.7270	13.4005	15.1647	0.8472	21.1412	8.8481
MSCNN [3]	0.7231	12.8203	13.9189	0.8102	17.5731	10.7991
DChP [4]	0.7060	11.5876	16.1415	0.8179	16.6215	9.9419
CAP [5]	0.7231	13.1945	13.3874	0.8364	19.0524	8.3102
DDN [6]	0.7383	10.9599	15.0212	0.8242	19.3767	9.5527
CycleGAN [7]	0.6490	13.6930	17.4213	0.5738	14.1578	16.4491
CycleDehaze [8]	0.6746	12.5412	15.7523	0.6923	15.8593	14.0566
Proposed Method	0.7411	13.8441	12.1098	0.8852	21.3012	6.2645

Table 2: Quantitative analysis of single image haze removal on synthetic outdoor hazy images.

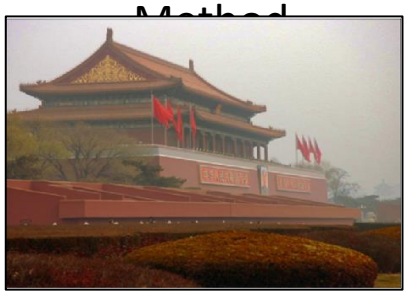
Method	SOHI (1000 images)		
	SSIM	PSNR	CIEDE2000
C2MSNet [1]	0.8929	21.6755	10.1637
DehazeNet [2]	0.8360	17.2575	12.0406
MSCNN [3]	0.7866	14.5844	17.1292
DChP [4]	0.7706	17.1029	22.0191
CAP [5]	0.8515	19.5641	14.5844
DDN [6]	0.8530	21.1962	13.0085
CycleGAN [7]	0.3832	12.0537	25.8382
CycleDehaze [8]	0.4613	12.5224	22.0191
Proposed Method	0.8919	22.9207	8.9262

Qualitative analysis

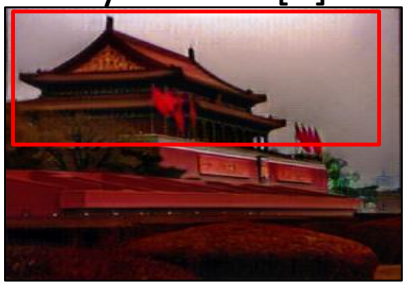
Hazy Image



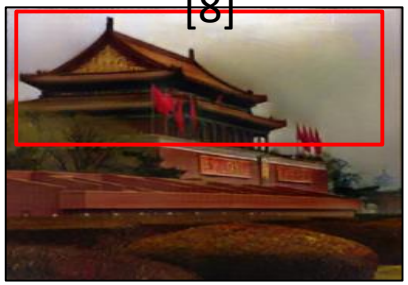
Proposed Method



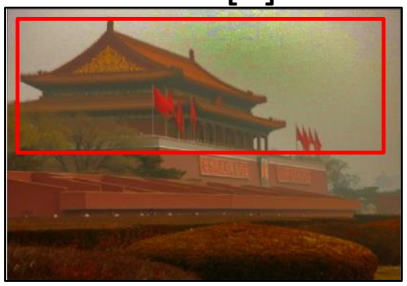
CycleGAN [7]



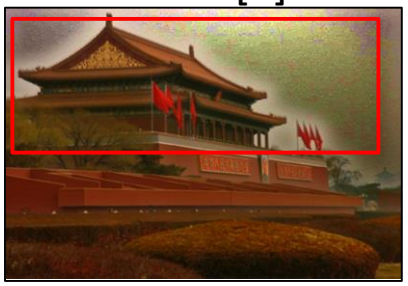
CycleDehaze [8]



DDN [6]



DChP [4]

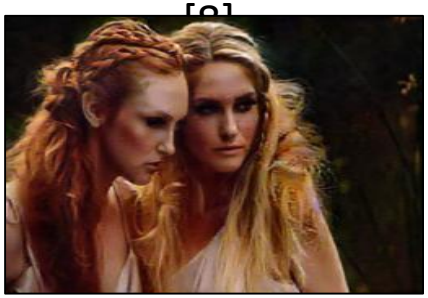


Hazy Image

Proposed Method

CycleGAN [7]

CycleDehaze [8]



DDN [6]

DChP [4]

Hazy Image

Proposed Method

CycleGAN [7]

CycleDehaze [4]

DDN [6]

DChP [4]

Conclusion

- The problem of unavailability of paired natural hazy images for the training of CNN has been solved using unpaired training approach.
- Color distortion in recovered images using existing end-to-end unpaired de-hazing architectures has been solved using an optical model-based approach.
- Experiments are carried out on three benchmark databases shows that proposed method outperforms other existing methods for single image haze removal.

Key references

1. Akshay Dudhane and Subrahmanyam Murala. 2018. C2MSNet: A Novel Approach for Single Image Haze Removal. In Winter Conference on Applications of Computer Vision (WACV). IEEE, 1397–1404.
2. Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. 2016. Dehazenet: An end-to-end system for single image haze removal. IEEE Transactions on Image Processing 25, 11 (2016), 5187–5198.
3. W. Ren, S. Liu, H. Zhang, J. Pan, X. Cao, and M.-H. Yang. Single image dehazing via multi-scale convolutional neural networks. In European Conference on Computer Vision, pages 154–169. Springer, 2016.
4. Kaiming He, Jian Sun, and Xiaoou Tang. 2011. Single image haze removal using dark channel prior. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 12 (2011), 2341–2353.
5. Qingsong Zhu, Jiaming Mai, and Ling Shao. 2014. Single Image Dehazing Using Color Attenuation Prior. In 25th British Machine Vision Conference (BMVC).
6. X. Yang, Z. Xu, and J. Luo. Towards perceptual image dehazing by physics-based disentanglement and adversarial training. In In Thirty third-second AAAI conference on Artificial Intelligence (AAAI-18), 2018.
7. J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 2242–2251. IEEE, 2017.
8. D. Engin, A. Genc, and H. Kemal Ekenel. Cycle-dehaze: Enhanced cyclegan for single image dehazing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 825–833, 2018.