Improving Generative Adversarial Networks with CapsuleNET

Arnab Karmakar

MLSP Course Project

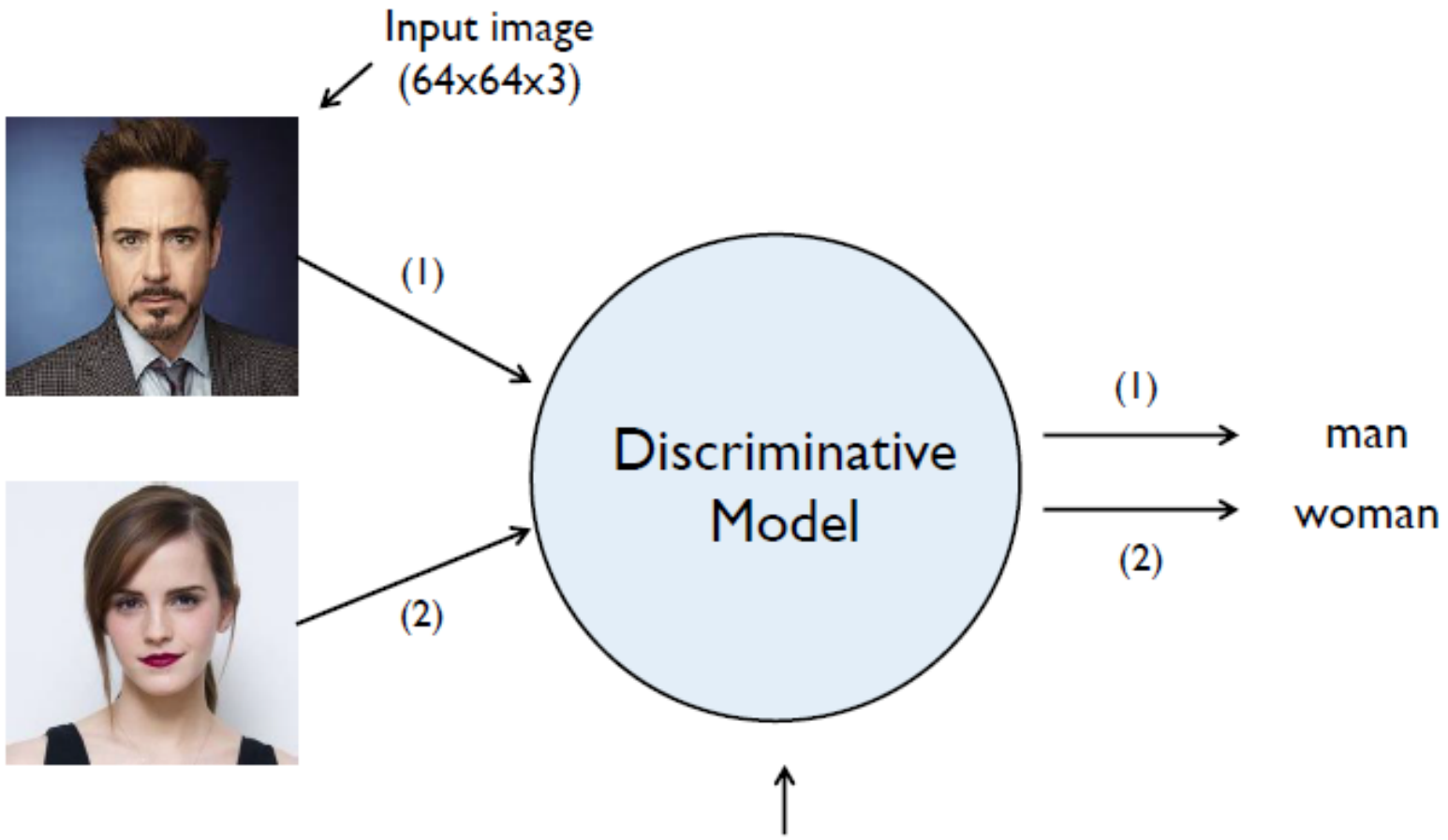
Indian Institute of Space Science and Technology

08.05.2018

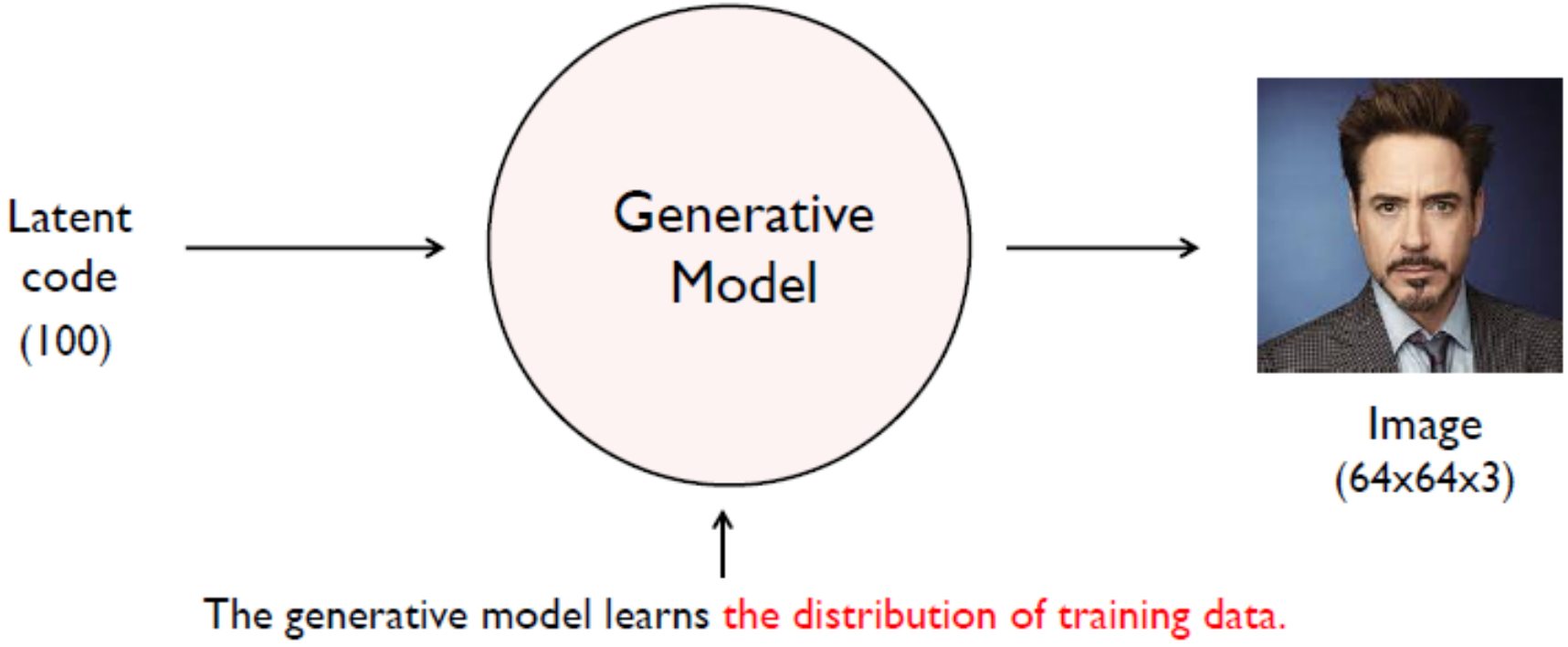
Project Objective

- Strong Discriminator – implementing CapsuleNET's high accuracy with less data principle
- Deep Convolutional Network as Generator – better quality of visual representation and the most stable generator model
- Developing a stable and accurate Generative Adversarial Network with a combination of DCGAN and CapsuleNET
- Faster Convergence with minimum time and resource
- We like to call it CapsuleGAN ☺

Discriminative Model



Generative Model

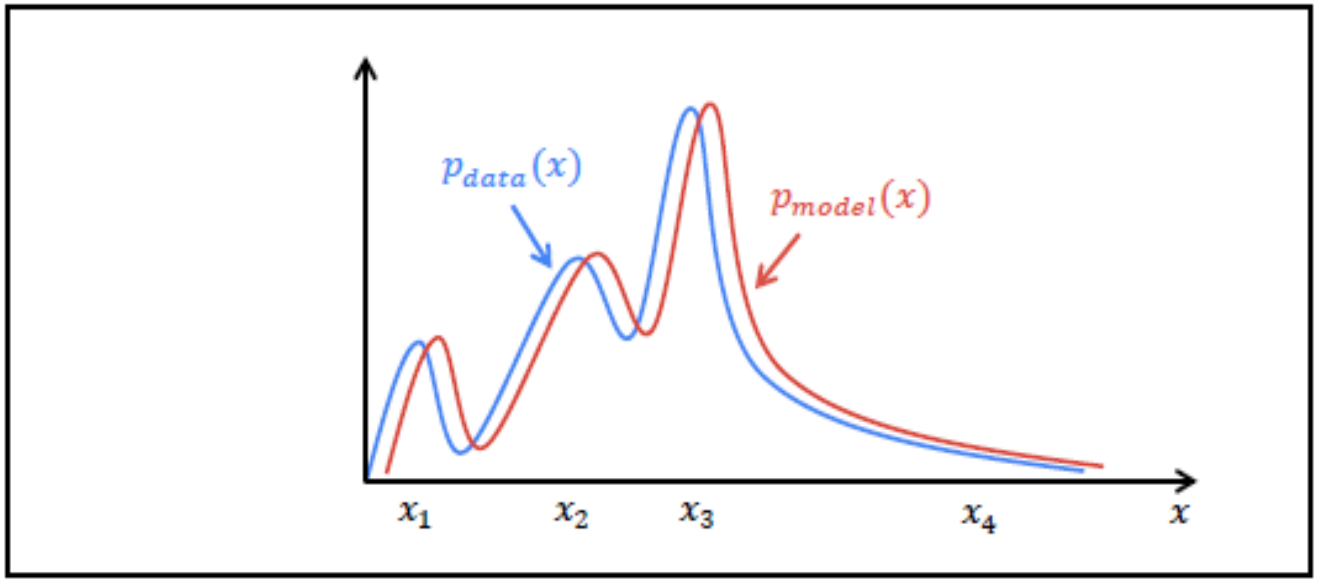


Intuition

Distribution of images generated by the model

The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

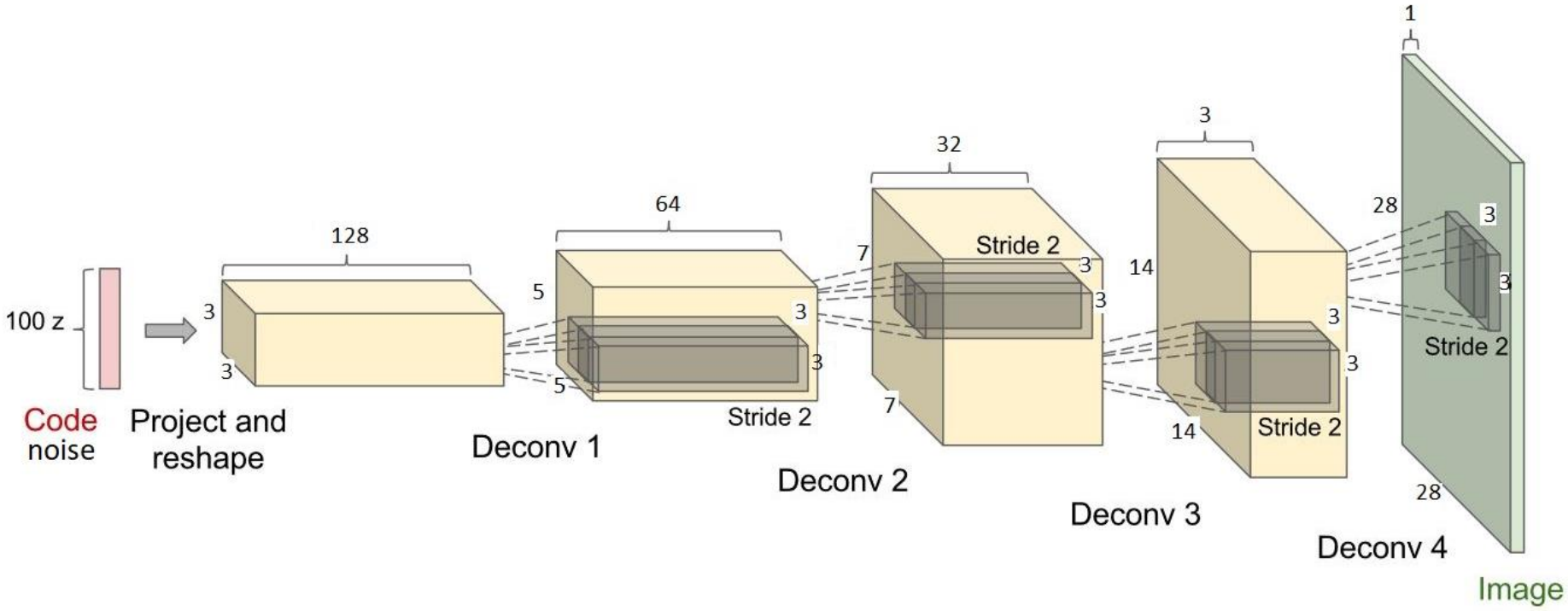
Distribution of actual images



GENERATOR

Architecture

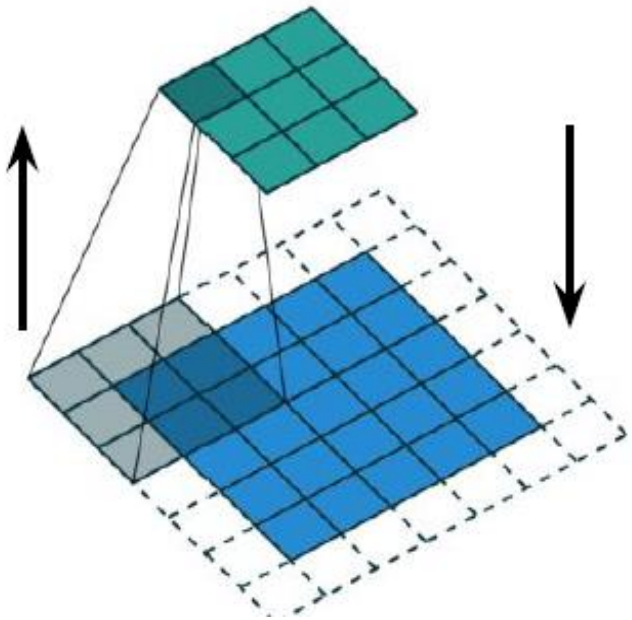
- Upsampling - backwards strided convolution
- 3x3 kernels throughout
- 128x(3x3) -> 64x(5x5) -> 32x(7x7) -> 3x(14x14) -> 1x(28x28)



Deconvolution: Transposed Convolution

Convolution:

(Up to) 3x3 blue pixels contribute to generate a single green pixel. Each of 3x3 blue pixels is multiplied by the corresponding filter value, and the results from different blue pixels are summed up to be a single green pixel.



Transposed-convolution:

A single green pixel contributes to generate (up to) 3x3 blue pixels. Each green pixel is multiplied by each of 3x3 filter values, and the results from different green pixels are summed up to be a single blue pixel.

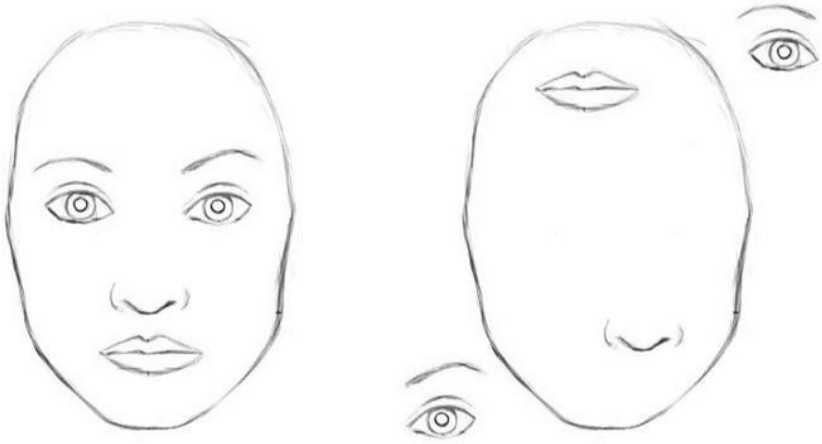
Why CNN as Generator

- Faster Learning
- Highly Stable
- Robust

DISCRIMINATOR

Problems with CNN

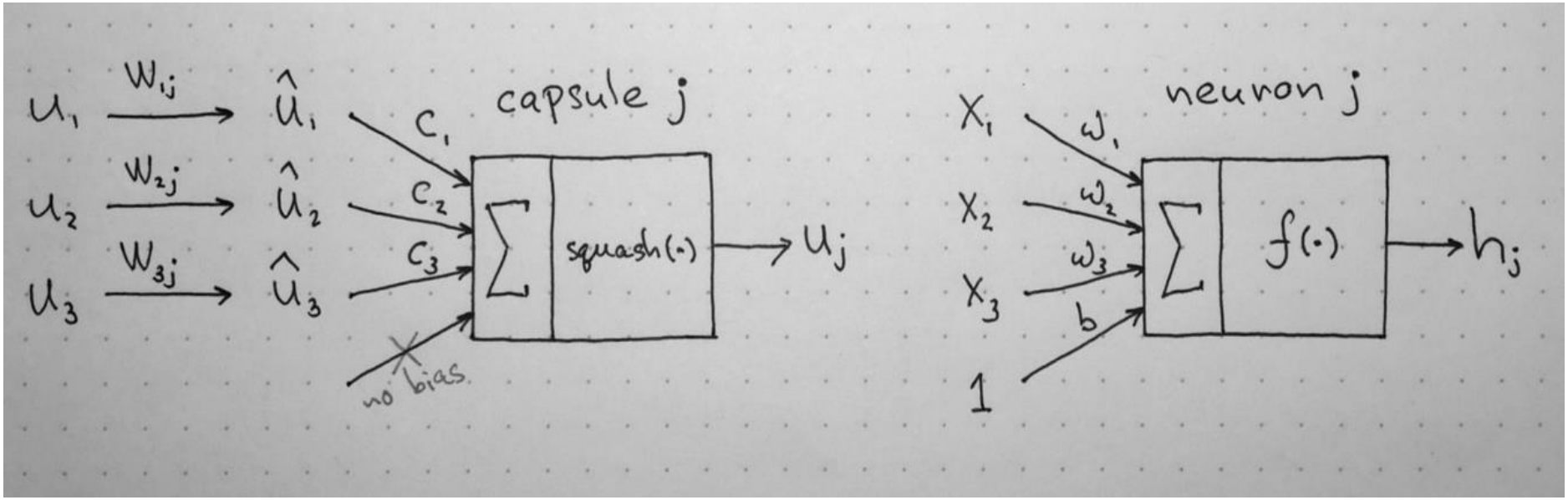
- Internal data representation of a convolutional neural network does not take into account important spatial hierarchies between simple and complex objects.
- Max Pooling reduce spatial size of the data flowing through the network - losing valuable information.



To a CNN, both pictures are similar, since they both contain similar elements

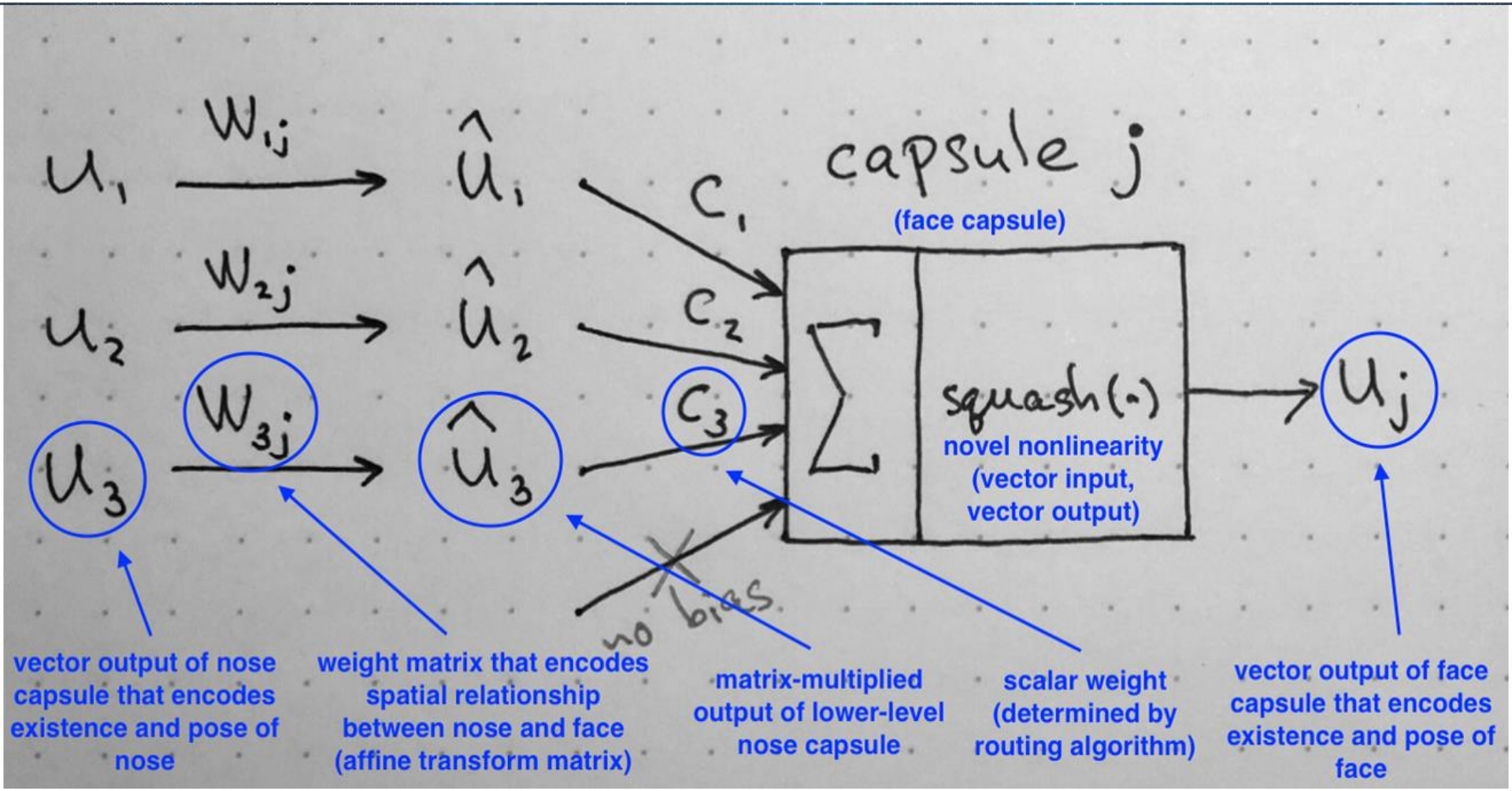
Capsule vs. CNN

- matrix multiplication of input vectors
- scalar weighting of input vectors
- sum of weighted input vectors
- vector-to-vector nonlinearity



CapsuleNET : example

lower level capsules detect eyes, mouth and nose respectively and out capsule detects face



CapsuleNET : example

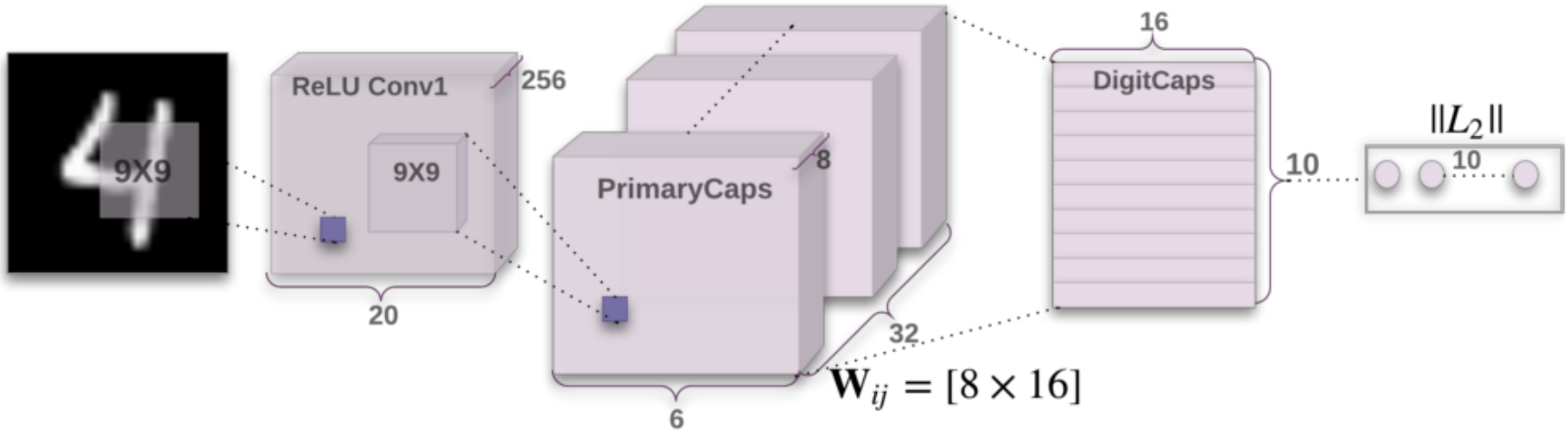
- Received input vectors (u1, u2 and u3) come from 3 other capsules in the layer below.
- Lengths of these vectors encode probabilities that lower-level capsules detected their corresponding objects and directions of the vectors encode some internal state of the detected objects
- Weight matrices W that encode important spatial and other relationships between lower level features (eyes, mouth and nose) and higher level feature (face)
- Multiplication by these matrices, gives the predicted position of the higher level feature. In other words, u1hat represents where the face should be according to the detected position of the eyes, u2hat represents where the face should be according to the detected position of the mouth etc.
- Scalar weight C is a measure of importance of the lower level feature to the higher level feature. (contribution towards final outcome of the neuron)
- So, in a sense, for each lower level capsule *i*, its weights *c_ij* define a probability distribution of its output belonging to each higher level capsule *j*.

CapsuleNET vs. CNN

Capsule vs. Traditional Neuron			
Input from low-level capsule/neuron		vector($\mathbf{u}_i$)	scalar(x_i)
Operation	Affine Transform	$\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$	–
	Weighting	$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Nonlinear Activation	$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$	$h_j = f(a_j)$
Output		vector($\mathbf{v}_j$)	scalar(h_j)

CapsuleNET architecture

Layer 1. Convolutional layer
Layer 2. PrimaryCaps layer
Layer 3. DigitCaps layer
Layer 4. Fully connected #1
Layer 5. Fully connected #2



CapsuleNET features

- **cut error rate by 45%** as compared to the previous state of the art.
- **only using a fraction of the data that a CNN would use**
- preserve hierarchical pose relationships between object parts - numerically as a 4D pose matrix.
- Translation, rotation, scale invariant.
- Capsules encapsulate all important information about the state of the feature they are detecting, in vector form.

Why CapsuleNET as Discriminator

- Stronger architecture than CNN – provides high accuracy
- Less time and resource consuming when used as a discriminator than as a generator
- Stable architecture
- Faster Convergence
- Slow when used as Generator

Why Training GANs are difficult

- GANs' instability may be that the generative distributions are weird, degenerate, and their support don't generally overlap with the true data distribution.
- Game theoretic view of seeking Nash equilibrium

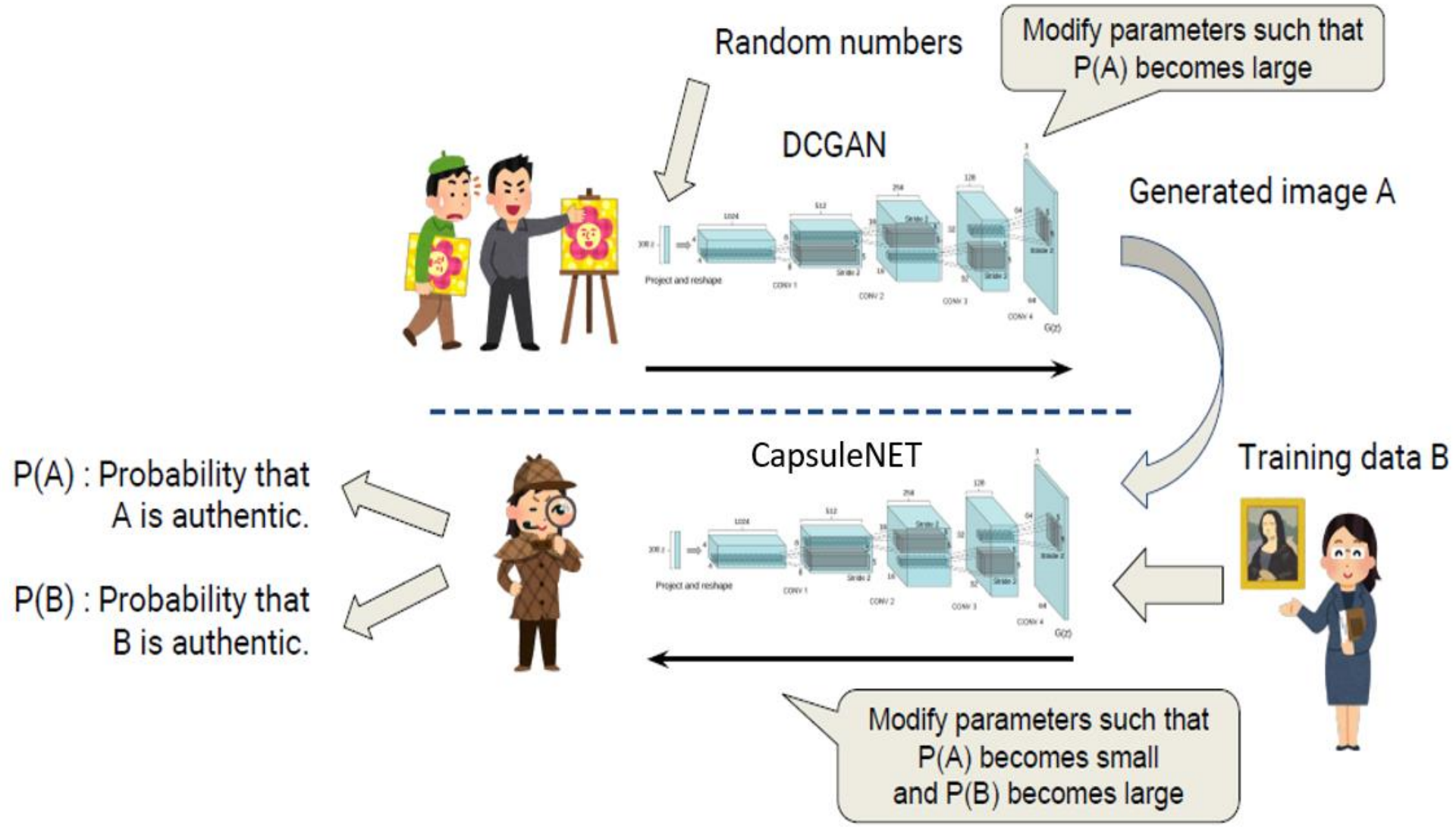
Crucially, the convergence of this algorithm relies on a few assumptions never really made explicit that don't always hold:

1. that the log-likelihood-ratio $\log \frac{q_\theta(y)}{p(y)}$ is finite, or
2. that the Jensen-Shannon divergence $\mathrm{JS}[q_\theta \| p]$ is a well-behaved function of θ

If q_θ and p have non-overlapping support, then

1. the log-likelihood-ratio and therefore KL divergence is infinite and not well defined
2. the Jensen-Shannon divergence is saturated so its maximum value:
3. the discriminator is extremely prone to overfitting:

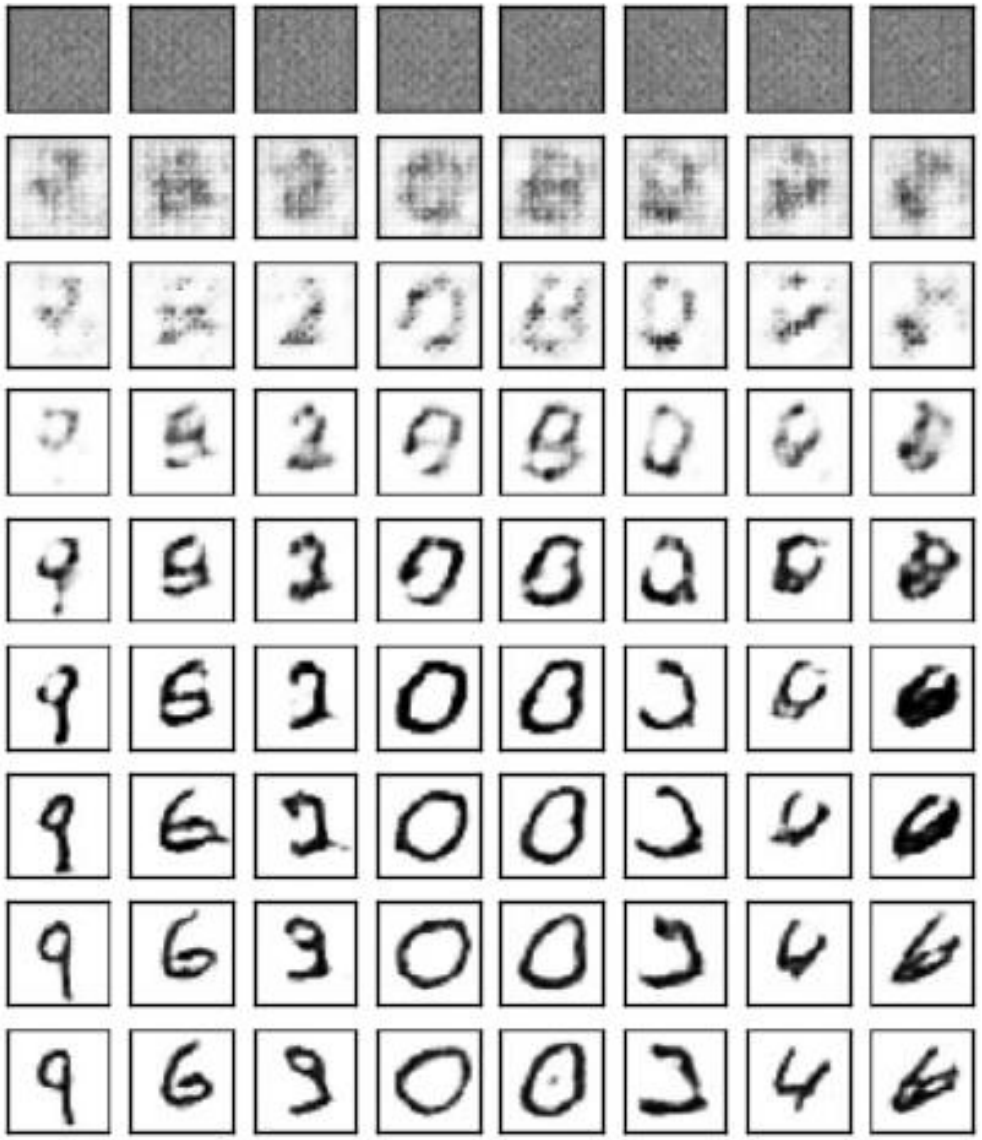
Training Loop of GAN



By repeating this loop, CNN becomes more accurate and GAN becomes more crafty

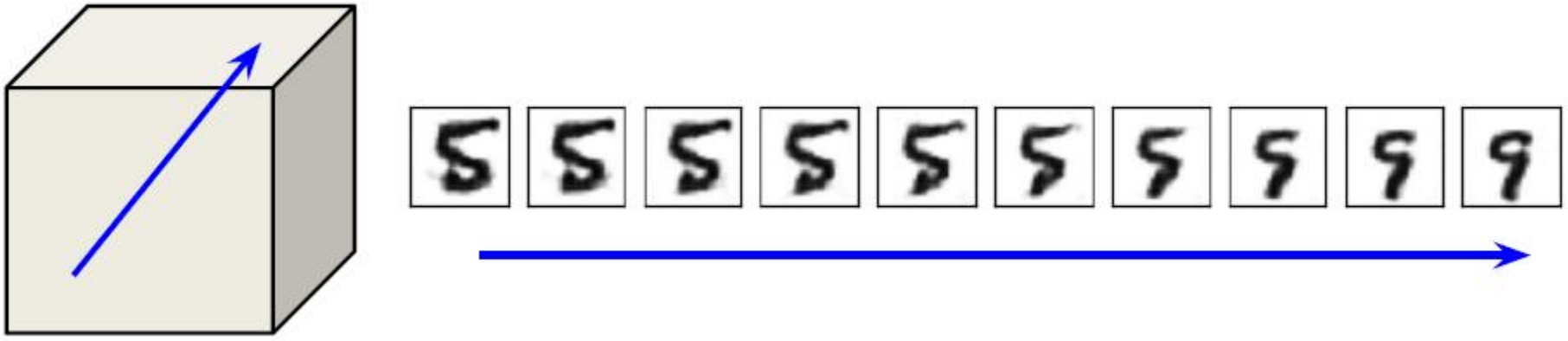
GAN Learning Process

Shows the evolution of images generated from the same input parameters during the training loop. (filters are initialized with random values.)

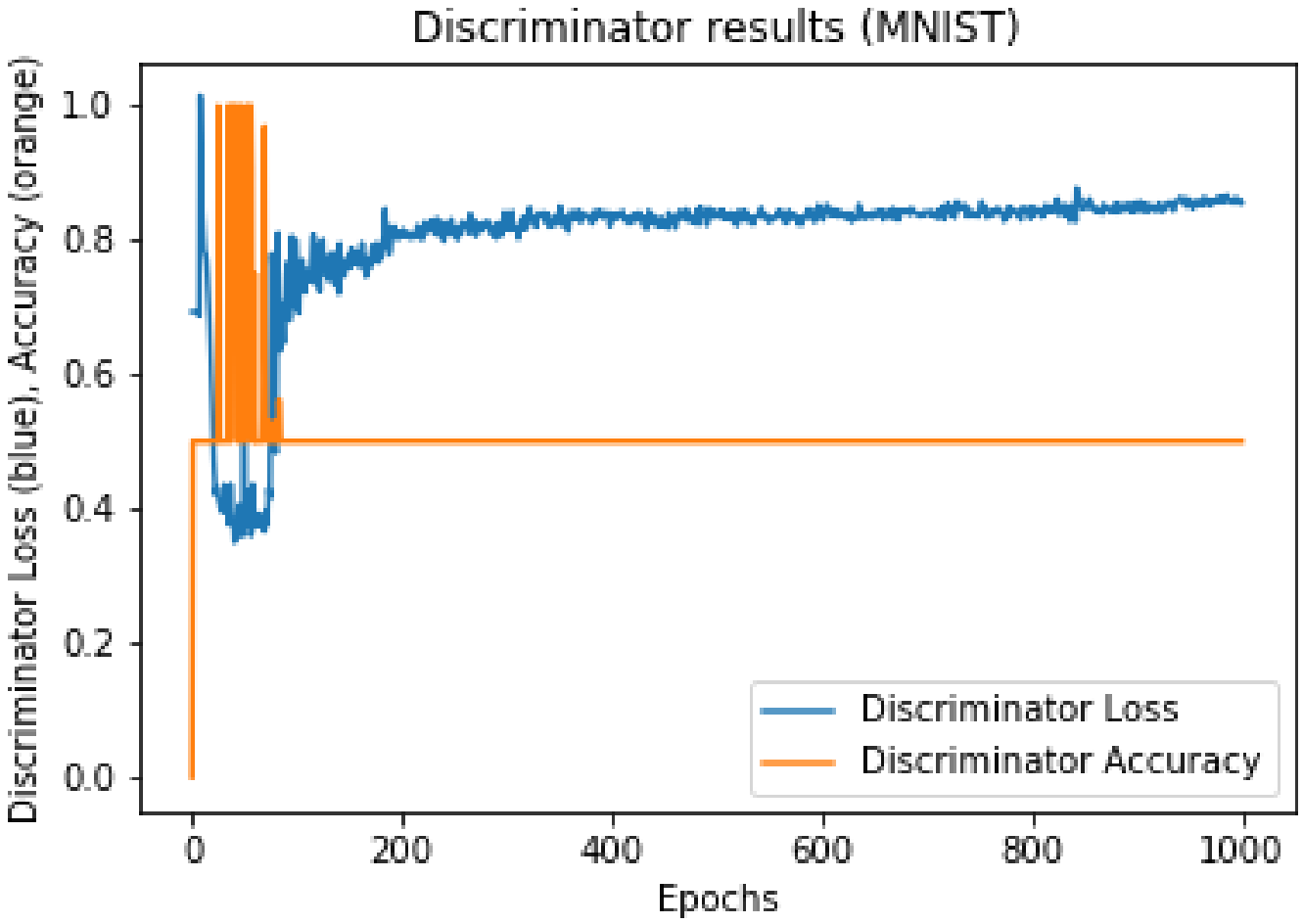


Playing with Input Parameters

- If we change the input parameter, the shape of generated image changes too. By making small, contiguous changes to the input, we can achieve a morphing effect.
- Since the input parameter is a point in the 28 dimensional space, we can draw a straight line between two points. The end points represent images before and after morphing.

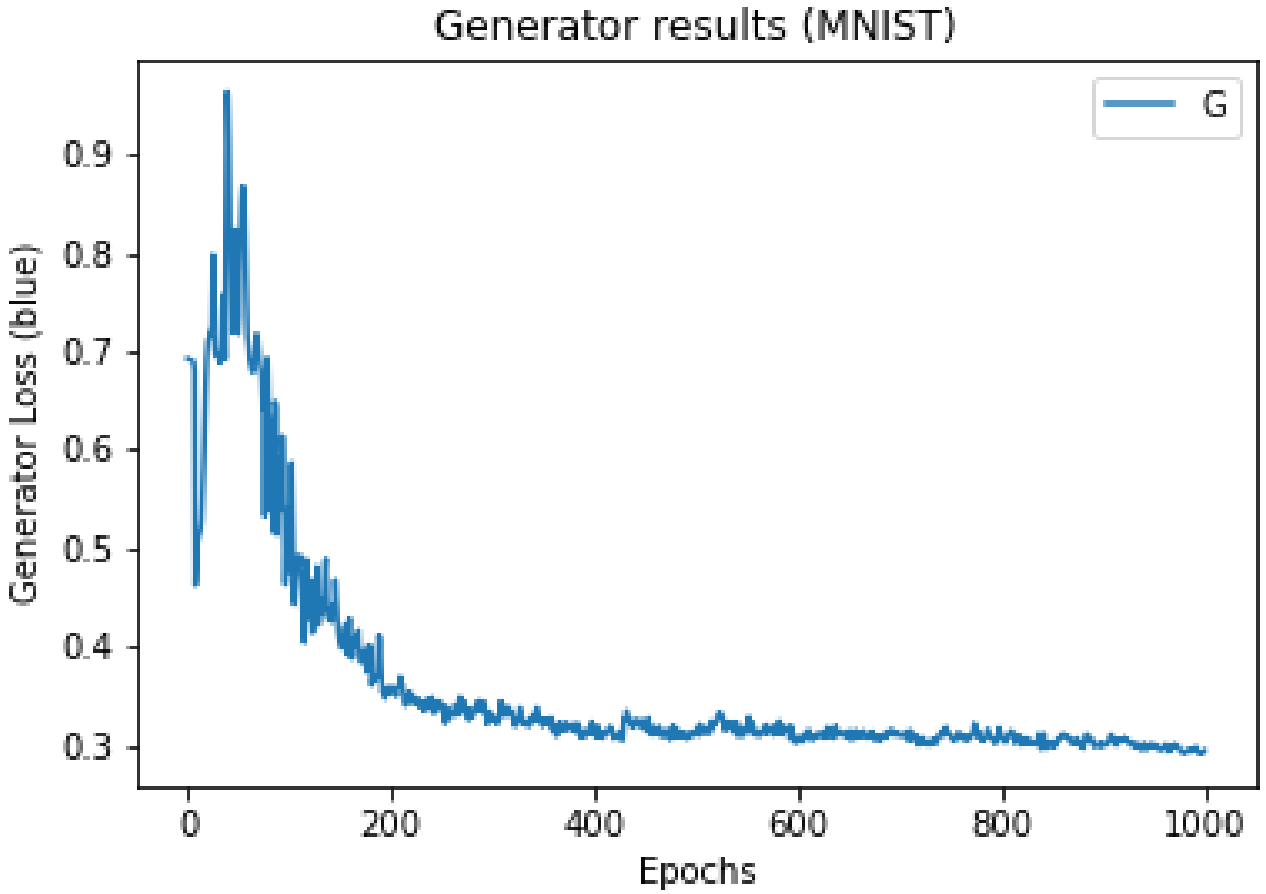


Results



Loss function: binary crossentropy

Results



Loss function: binary crossentropy

Implications

- Discriminator accuracy starts from zero, goes to maximum (100%) quickly, implies Discriminator's learning is superior than Generator.
- Strong Discriminator – Accuracy goes up in the initial stage, loss goes down…. Generator loss goes high simultaneously.
- Discriminator accuracy fluctuates – Generator starts learning, Generator loss starts to fall
- Discriminator accuracy at standstill (50%), reached convergence
- Generator Loss keep on falling, learning better and better parameters to mimic actual data
- Generator loss, Discriminator loss and accuracy at standstill, reached convergence – weight updation still going on but on a very minute scale, visual representation of generated data (sharpness, contrast) increases slightly.

Implications

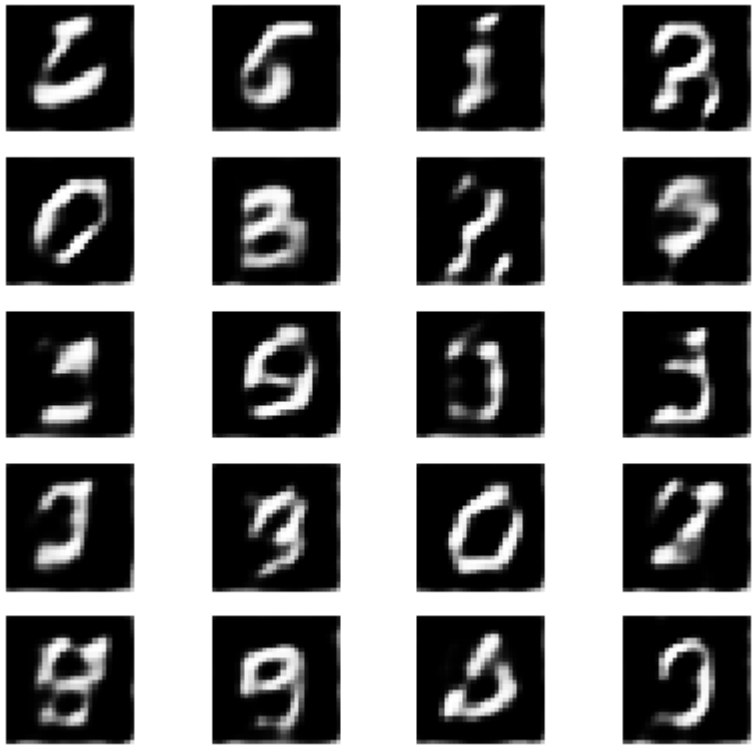
- Discriminator Accuracy never goes below 50% due to the fact that each batch consists of equal number of generated as well as real world images – strong discriminator is helpful
- Also implies that Discriminator's learning capacity is very high.

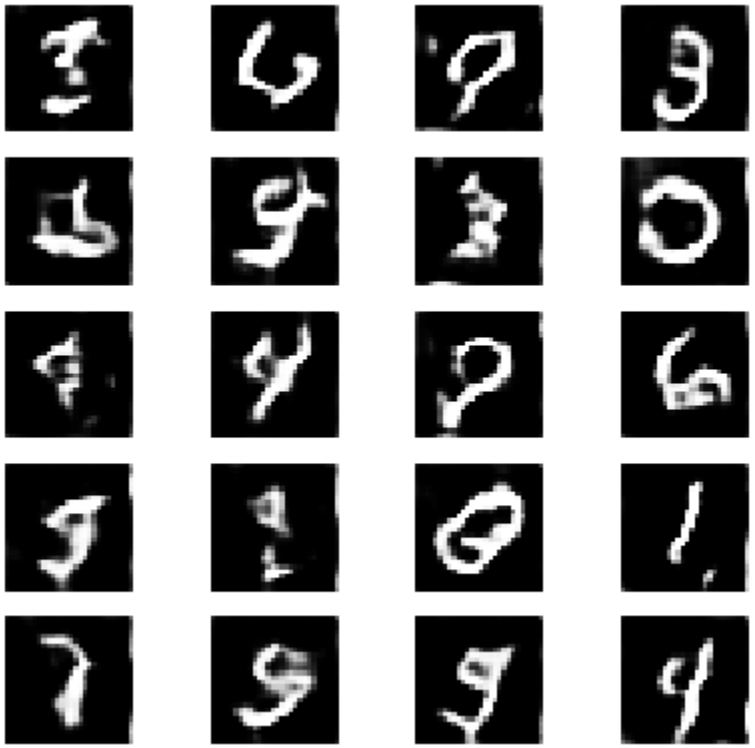
Contribution and Comparison

- One of the first experimentation with GANs using CapsuleNET
- Convergence within 1/3 number of the training epochs compared to DCGAN

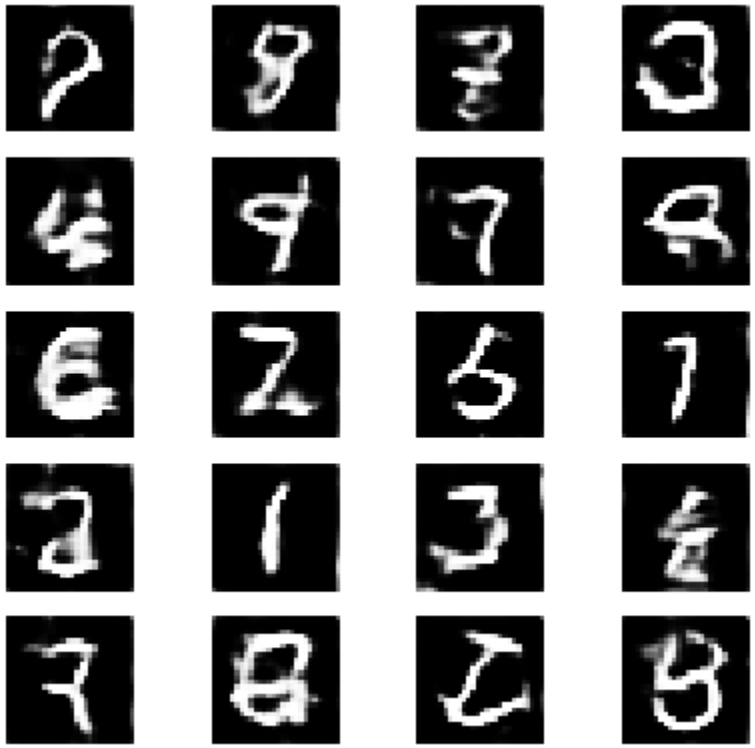
  (DCGAN uses CNN as both discriminator and generator)
- Results are highly satisfactory

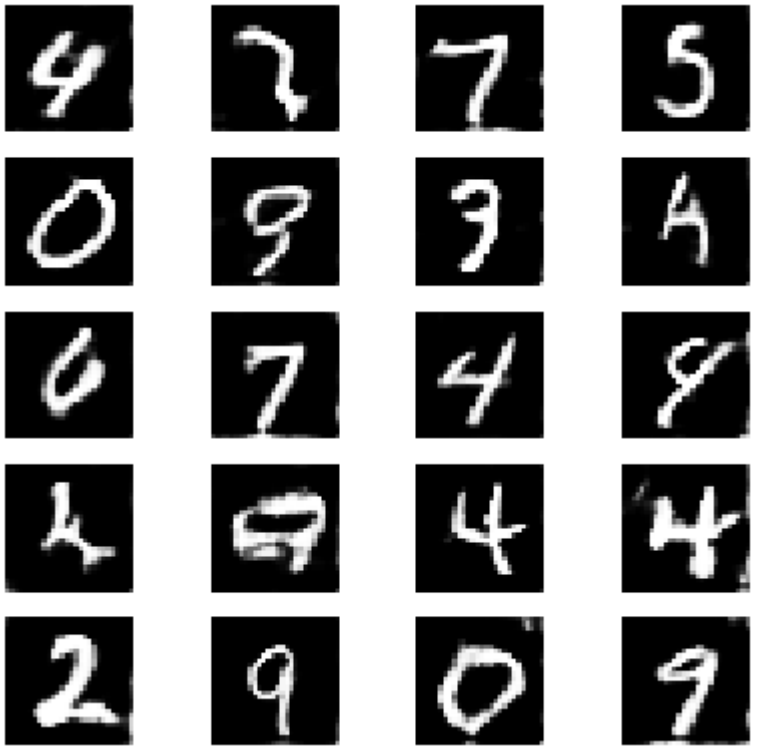
DCGAN	CapsuleGAN
1000 epochs	1000 epochs





DCGAN	CapsuleGAN
10000 epochs	10000 epochs





Future Scope

- Robustness to be tested on other datasets
- Generation of RGB images (3 channel) to be tested
- Using CapsuleNET as both discriminator and generator which satisfies time constraints

References

- https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b
- http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/
- Deep Learning – Goodfellow, LeCun, Carter

Thank You

- Questions?
- Suggestions?