Deep Structured Models #2: Deep (Convolutional) Neural Networks

Member: Qilin Li, Xinjie Lei, Hongnian Yu, DiJia Su

May 15, 2017

Learning goes deep

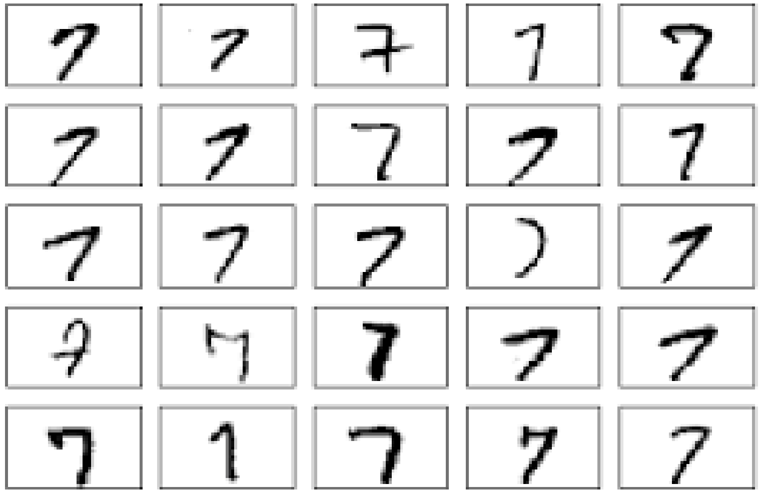
- How do Structured Models go to “deep”?
- Answer: One way is introducing CNN to traditional method.

- Here we review two papers, both are generative models, incorporating CNN with traditional methods

Review of VAE
(Variational Auto-Encoder)

Background: VAE

- Example:
  Suppose you have a handwriting image dataset, number 7



- Want to generated new images **similar** to the ones in the data-set.

- Use Variational Auto-Encoder (VAE)
  - doesn‘t require strong assumption of input data
  - Computational efficient

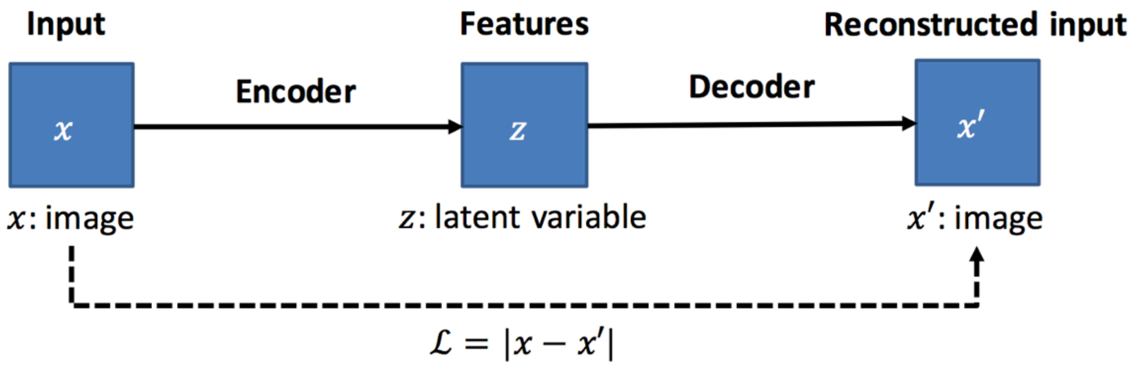
Background: VAE

- Formally:
- VAE is unsupervised learning. It is a generative model
- Input: X (images)
  - X distributed according to some unknown distribution P(X)
- Goal: learn the P(X)
  - Captures the dependencies between pixels
- Output: a distribution P’(X) that’s close to P(X)

Background VAE

- Two components:
  - Encoder:
    - Convert x to latent (hidden) variables z that lives in high dimensional space Z
    - $\boldsymbol{x} \rightarrow \boldsymbol{z}$
  - Decoder:
    - convert latent variables z to x'
    - Hopefully x' close to x
    - $\boldsymbol{z} \rightarrow \boldsymbol{x}'$

- Basic pipeline



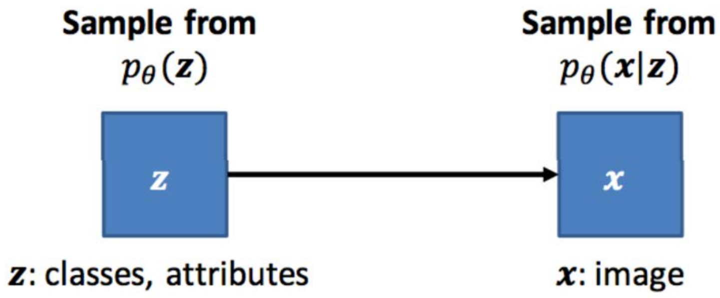
Background VAE

- Maximum Likelihood – try to maximize the following:

- Objective: $P_\theta(x) = \int_z P_\theta(x,z)dz = \int_z P_\theta(x|z)\, P_\theta(z)dz$

- θ is the parameter of the model

- $P_\theta(z)$ is very complicated distribution
  - Contains many dependencies…
    - Angle of the digits, the width of stroke, stylish attributes



- $P_\theta(x|z)$ is probability of x given z

- Problem: solving this equation directly is intractable

Alternative Solution

- Since directly maximizes $P_\theta(x) = \int_z P_\theta(x|z)\, P_\theta(z) dz$ is intractable, what can we do ?

- Work around:
  - rewrite $P_\theta(z|x) = \frac{P_\theta(x|z) P_\theta(z)}{P(x)}$ (Bayes rule)
  - Make approximation:
    - Let $q_\phi(z|x) \cong P_\theta(z|x)$
      - q_ϕ takes input x and gives a distribution of z that are **likely to produce P(x)**
      - Hopefully (usually) lives in a lower dimensional space

KL Divergence

- Introduce the KL divergence between q and p distributions:
  - $KL(q||p) = \int q(t) \log \frac{q(t)}{p(t)} \, dt = E_q(\log q - \log p) = E_q(\log q) - E_q[\log p]$
  - It’s a measure the similarity between two distribution
  - Important property:
    - $KL(q||p) \geq 0$
    - $KL(q||p) = 0 \;\leftrightarrow\; p = q$

KL Divergence

- Let's take the KL between $q_\phi(z|x)$ and $p_\theta(z|x)$
- Recalled that $q_\phi(z|x) \cong P_\theta(z|x)$
  - $KL(q_\phi(z|x)||p_\theta(z|x)) = \mathrm{E}_{q_\phi(z|x)}[\log q_\phi(z|x) - \log P_\theta\ (z|x)]$
  - After some manipulation, you get:
  - $\log P_\theta(x) - KL(q_\phi(z|x)||p_\theta(z|x)) = \mathrm{E}_{q_\phi(z|x)}[\log P_\theta\ (x|z)] - KL(q_\phi(z|x)||p_\theta(z))$

KL Divergence

- Let's take the KL between $q_\phi(z|x)$ and $p_\theta(z|x)$
  - $KL(q_\phi(z|x)||p_\theta(z|x)) = \mathrm{E}_{q_\phi(z|x)}[\log q_\phi(z|x) \; - \log P_\theta \; (z|x)]$
  - After some manipulation, you get:
  - $\log P_\theta(x) \; - KL(q_\phi(z|x)||p_\theta(z|x)) = \mathrm{E}_{q_\phi(z|x)}[\log P_\theta \; (x|z)] \; - KL(q_\phi(z|x)||p_\theta(z))$
  - This is our objective function in final form!

KL Divergence

- Original Objective: $\boxed{P_\theta(x)} = \int_z P_\theta(x,z)dz = \int_z P_\theta(x|z)\, P_\theta(z)dz$ **(Intractable)**

- Final Objective function:
- $\boxed{\log P_\theta(x)} - KL(q_\phi(z|x)||p_\theta(z|x)) = \mathrm{E}_{q_\phi(z|x)}[\log P_\theta\,(x|z)] - KL(q_\phi(z|x)||p_\theta(z))$

- Core equation of VAE

- Note: by maximizing equation above, we maximize $P_\theta(x)$, while simultaneously and magically "pulling" $q_\phi(z|x)$ closer to $p_\theta(z|x)$

ELBO

- ELBO (Evidence Lower Bound Objective)
- $ELBO = \mathcal{L}_{VAE} = \mathrm{E}_{q_\phi(z|x)}[P_\theta\ (x,z)] - \mathrm{E}_{q_\phi(z|x)}[q(z|x)]$
- Rewrite final objective as:
- $\log P_\theta(x) = KL(q_\phi(z|x)||p_\theta(z|x)) + \mathcal{L}_{VAE} \geq \mathcal{L}_{VAE}$
    - Since KL >= 0

Training

- For training, we use the following form:

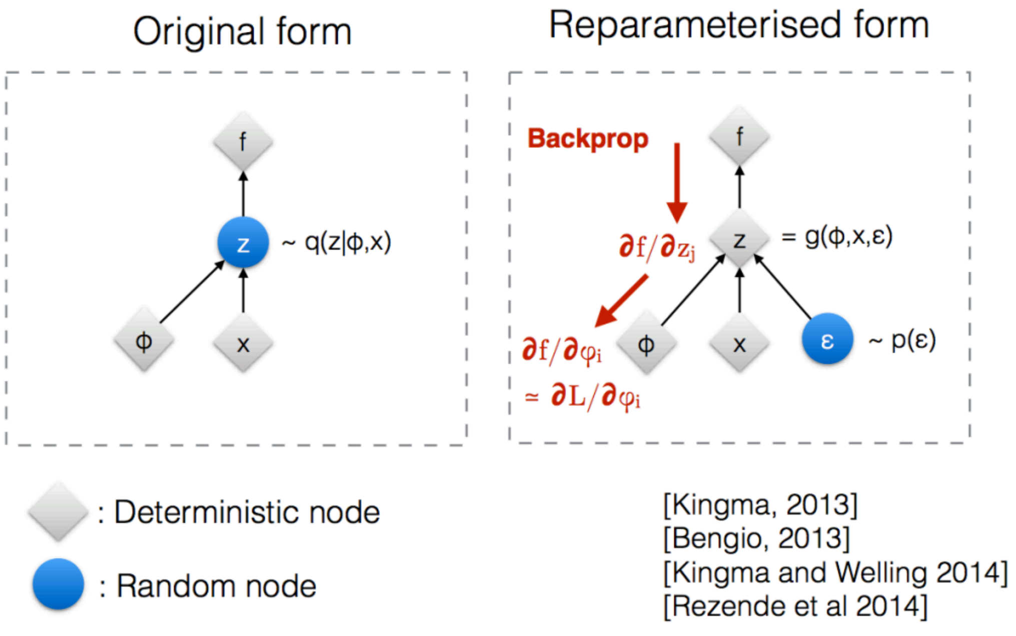
- $\mathcal{L}_{VAE}(x;\phi,\theta) = -KL(q_{\phi}(z|x)\big|\big|p_{\theta}(z|x)\big) + \mathrm{E}_{q_{\phi}(z|x)}[log p_{\theta}(x|z)]$

  Regularizer Reconstruction

- For real implementation, need to use backpropagation (which require re-parametrization trick)

Trick

- Re-parametrization Trick:
- Backpropagation is not possible through random sampling!
- Sampling Trick:



- Write $z^{(i,l)} \sim N(\mu^{(i)}, \sigma^{2(i)})$

$$z^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \varepsilon_i \quad \varepsilon_i \sim N(0,1)$$

where $g(.)$ is a deterministic differentiable function which maps $z^{(i,l)}$ to the latent distribution.

Empirical

- With trick, we can rewrite:

- $\mathcal{L}_{VAE}(x;\phi,\theta) = -KL(q_\phi(z|x)||p_\theta(z|x)) + \mathrm{E}_{q_\phi(z|x)}[logp_\theta(x|z)]$

  with

- $\mathrm{E}_{q_\phi(z|x)}[logp_\theta(x|z)] \cong \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(x|z^l),$

Empirical

- Empirical objective of the VAE with Gaussian latent variables:

$$\tilde{L}_{VAE}(x;\phi,\theta) = -KL(q_\phi(z|x)||p_\theta(z)) + \frac{1}{L}\sum_{i=1}^{L} \log \ p_\theta(x|z^l)$$

Regularizer

Reconstruction

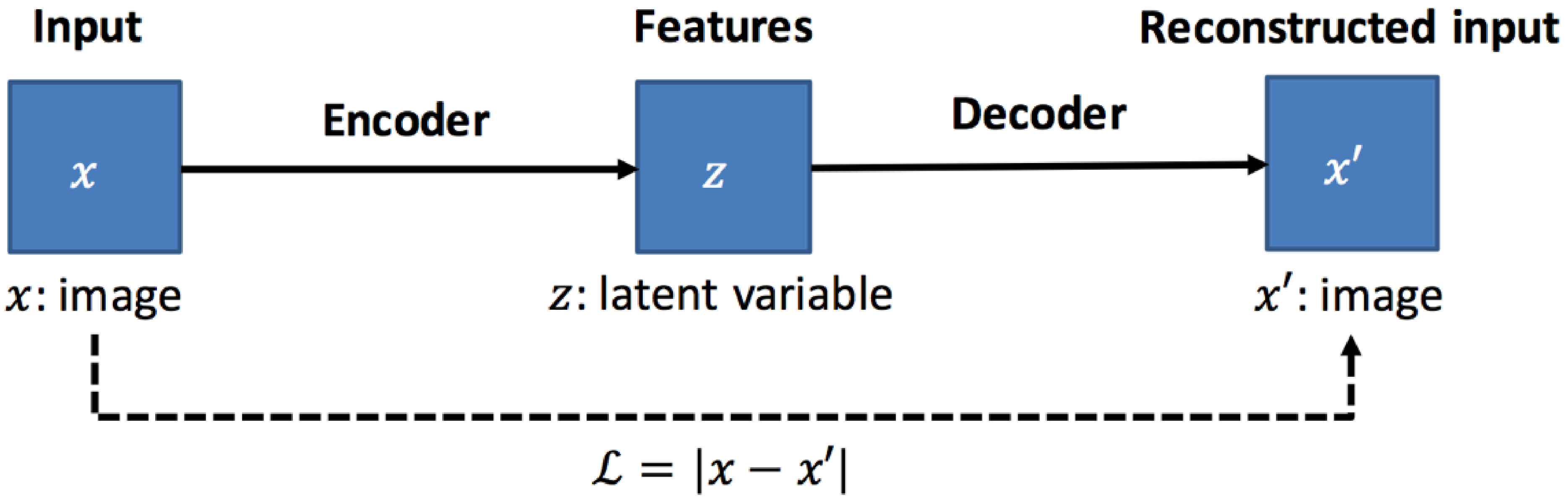
where

1) $z^l = g_\phi\left(x, \epsilon^{(l)}\right), \epsilon^{(l)} \sim N(0, I)$

2) q_ϕ(z|x) is reparametrized with a deterministic, differentiable function g_ϕ

VAE (Summary)

- Basic pipeline



- $Goal: \ \ Maximize \ P_\theta(x)$

Empirical

- Empirical objective of the VAE with Gaussian latent variables:

$$\tilde{L}_{VAE}(x;\phi,\theta) = -KL(q_\phi(z|x)||p_\theta(z)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(x|z^l)$$

where

1. $\tilde{L}_{VAE}$ guarantees the lower bound of $p_\theta(x)$.
2. $p_\theta(z)$ is the prior distribution of a very complicate latent variable z
3. $z^l = g_\phi\left(x, \epsilon^{(l)}\right), \epsilon^{(l)} \sim N(0, I)$
4. q_ϕ(z|x) is reparametrized with a deterministic, differentiable function g_ϕ
5. Maximize via SGD

Learning Structured Output Representation using Deep Conditional Generative Models

By Kihyuk Sohn Xinchen Yan Honglak Lee

2015 NIPS

Structure

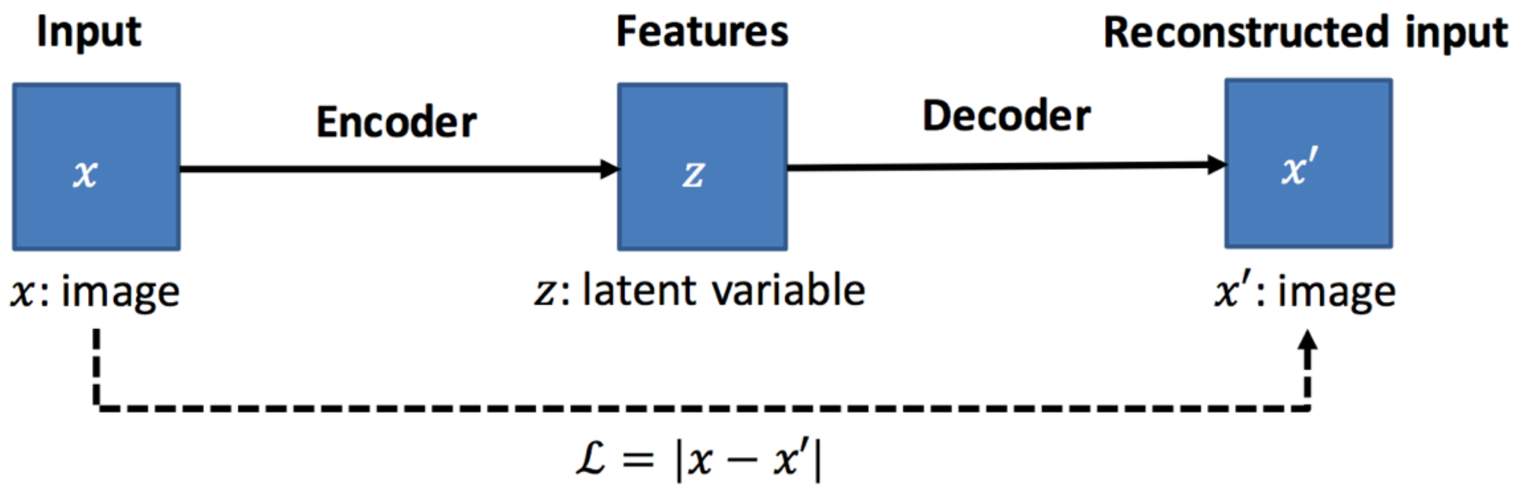
- Motivation
- Theory background
  - CVAE: Conditional Variational Autoencoder
  - CGM: Conditional Generative Model
  - Recurrent CVAE
- Model setup
- Other Tricks
  - GSNN
  - Image processing
- Evaluation
- Test results
- Summary

Motivation

- CNN is not suitable for modeling a distribution with multiple modes, e.g. mapping from single input to many possible outputs.
- Exact distribution $p_\theta(z|x)$ is intractable.

Theory background
CVAE

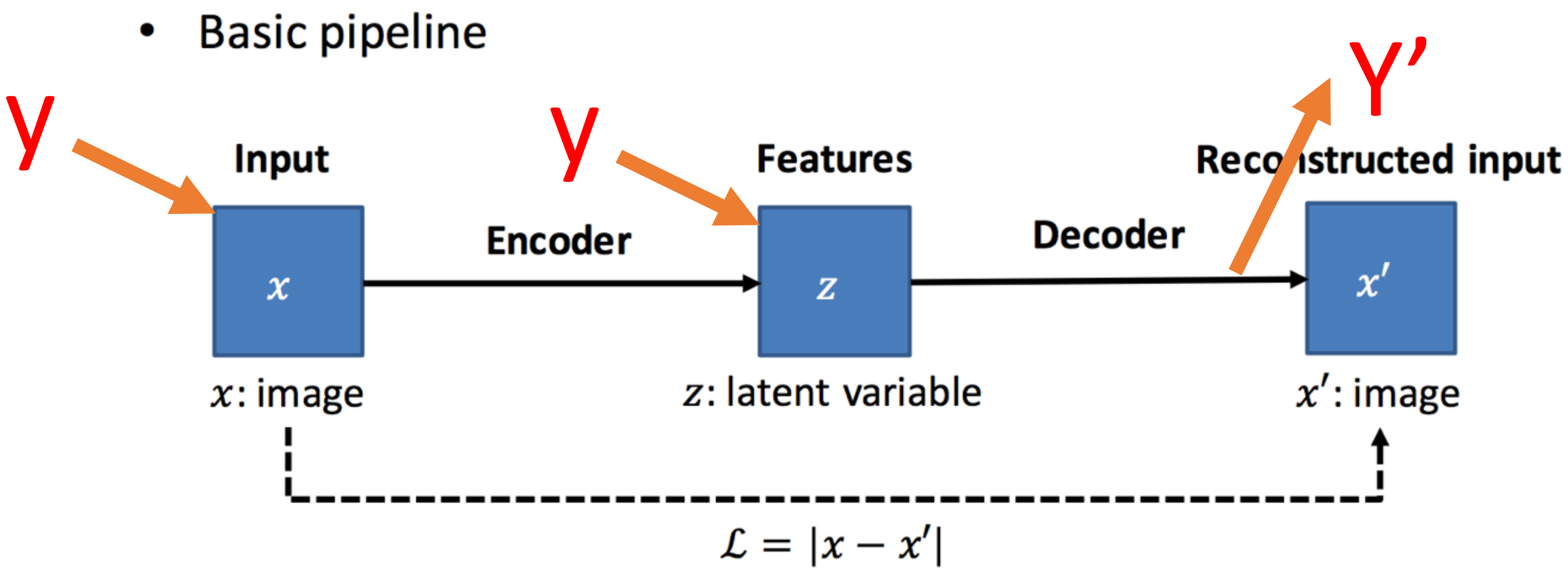
- We already know that, for traditional VAE, it is an autoencoder, unsupervised method.
    - Basic pipeline



- Question? How about we have output label y?

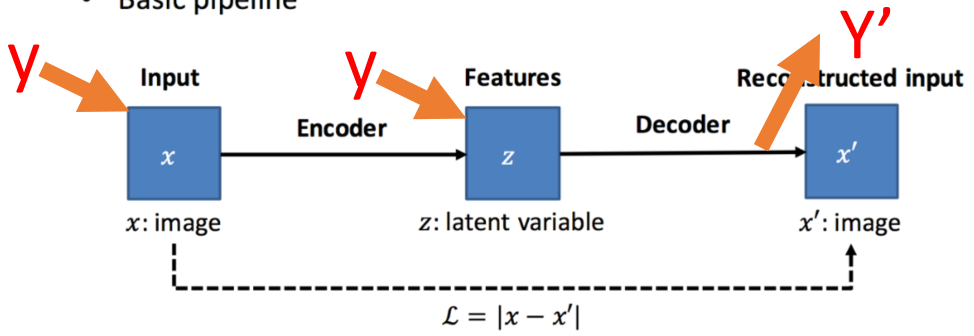
Theory background
CVAE

- Introduce y into VAE model, such as image label .
- The conditional variational autoencoder (CVAE) has an extra input to both the encoder and the decoder.



Theory background
CVAE

- Basic pipeline



- Introduce y into VAE model.
- Modify the object function
- from VAE to CVAE.

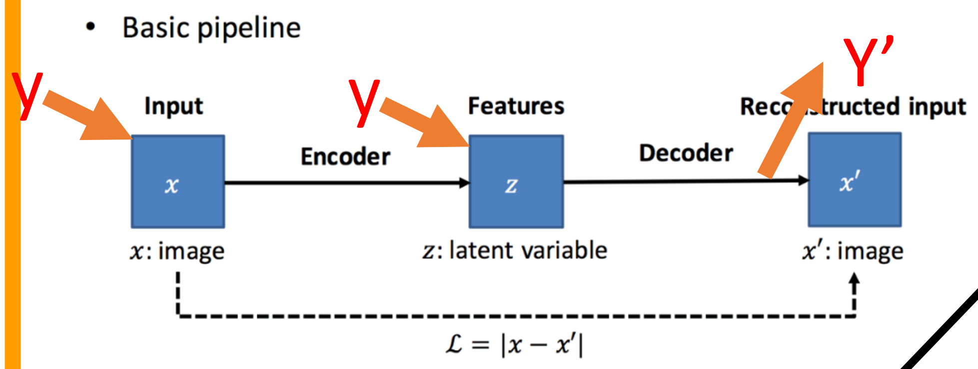
- Original VAE version:
- $\log P_\theta(x) \geq \mathbf{E}_{q_\phi(z|x)}[\log P_\theta(x|z)] - \boldsymbol{KL}(q_\phi(z|x)||p_\theta(z))$
  - (reminder: RHS is the ELBO: Evidence Lower Bound Objective)
- Modified CVAE version:
- $\log P_\theta(x|y) \geq \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta(x|y,z)] - \boldsymbol{KL}(q_\phi(z|x,y)||p_\theta(z|y))$

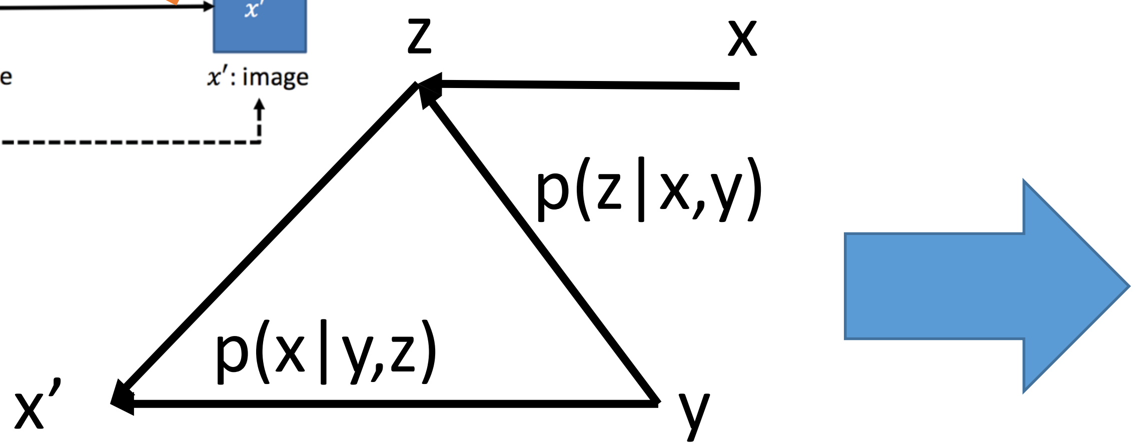
- Mathematically the variational approach is the same as above except with conditioning on y. That is way we call it CVAE.

- Training target: maximize target function. (Maximize conditional log likelihood)

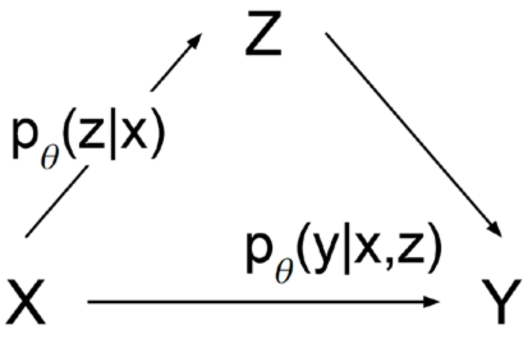
Theory background
CVAE as deep-CGM

- However, if we want to predict y instead of decoding x, we have to modify CVAE to Conditional Generative Model (CGM), which has y as output.





CVAE (reconstruction)



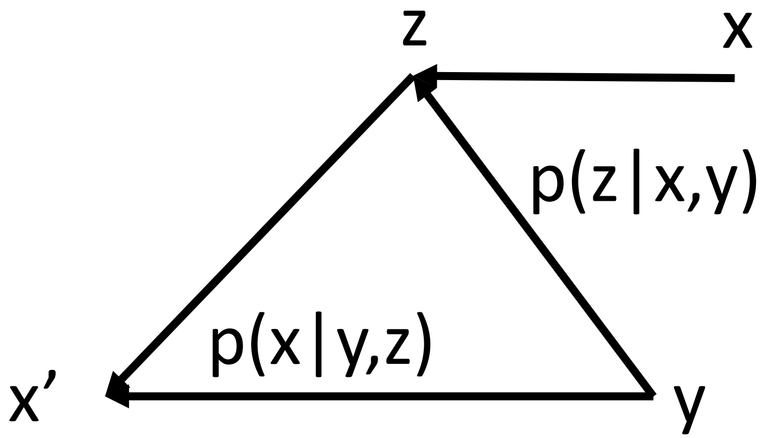
CGM (generation)

Theory background
CVAE as deep-CGM

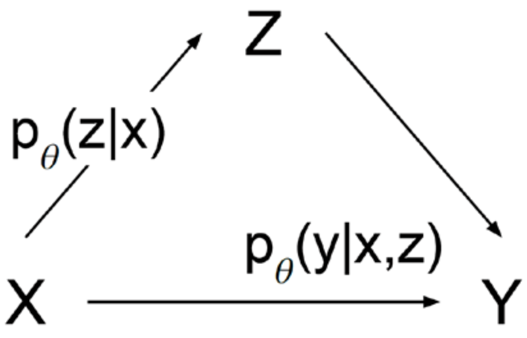
- Modify the object function for CGM (Conditional Generative Model)

- CVAE version:
- $\log P_\theta(x|y) \geq \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta\,(x|y,z)] - \boldsymbol{KL}(q_\phi(z|x,y)||p_\theta(z|y))$
- CVAE as CGM version:
- $\log P_\theta(y|x) \geq \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta\,(y|x,z)] - \boldsymbol{KL}(q_\phi(z|x,y)||p_\theta(z|x))$

- Just swop x, y



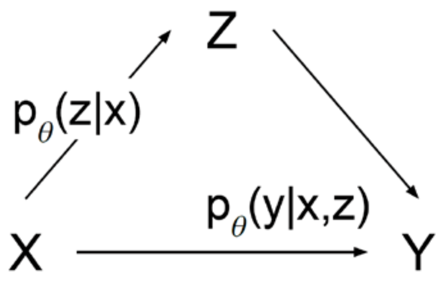
CVAE (reconstruction)



CGM (generation)

Theory background
CVAE as deep-CGM

- Understanding CVAE as CGM (Conditional Generative Model)
- Three variables: Input variables x, output variables y, and latent variables z.
- For given observation x, z is drawn from the prior distribution $p_\theta(z|x)$, and the output y is generated from the recognition distribution $P_\theta(y|x, z)$.



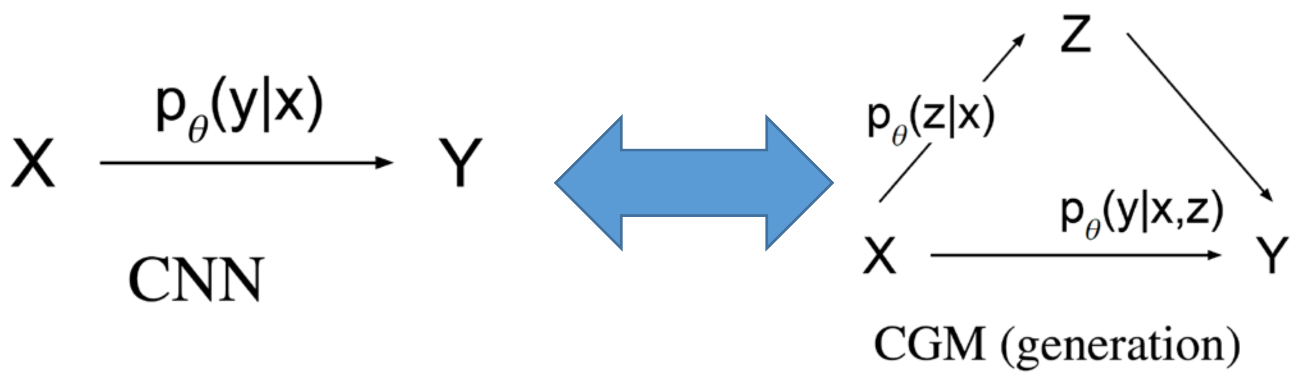
CGM (generation)

CVAE as CGM version:

$$\log P_\theta(y|x) \geq \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta\,(y|x,z)] - \boldsymbol{KL}(q_\phi(z|x,y)||p_\theta(z|x))$$

Theory background
CVAE as deep-CGM

- Understanding CVAE as CGM (Conditional Generative Model)
- Compared to the baseline CNN, the latent variables z allow for modeling multiple modes in conditional distribution of output variables y given input x, making the proposed CGM suitable for <u>modeling one-to-many mapping</u>.

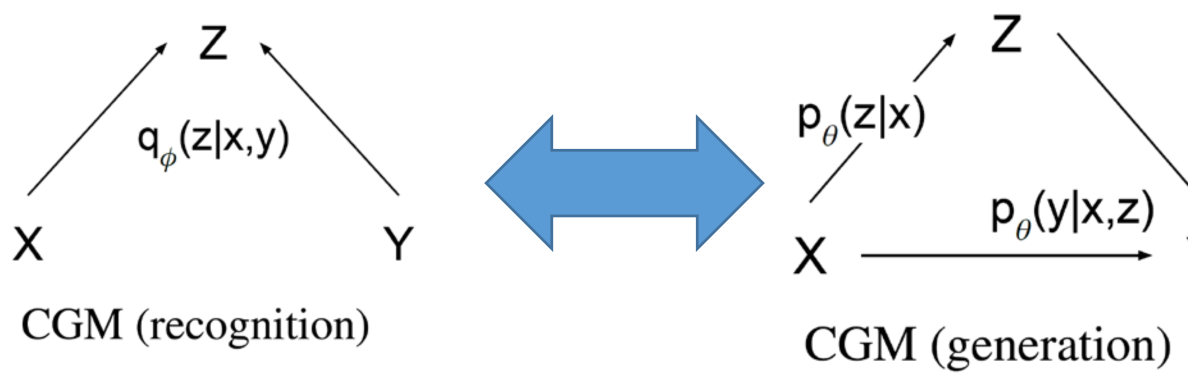


CVAE as CGM version:

$$\log P_\theta(y|x) \geq \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta(y|x,z)] - \boldsymbol{KL}(q_\phi(z|x,y)||p_\theta(z|x))$$

Theory background
CVAE as deep-CGM

- Understanding CVAE as CGM (Conditional Generative Model)
- Recognition” model: $q_\phi(z|x,y)$ is introduced to approximate the true posterior $P_\theta(z|x,y)$.
- Usage: $P_\theta(y|x,z)$ can be approximated by drawing samples $z^{(l)}$ ($l = 1, \ldots, L$) by the recognition distribution $q_\phi(z|x,y)$

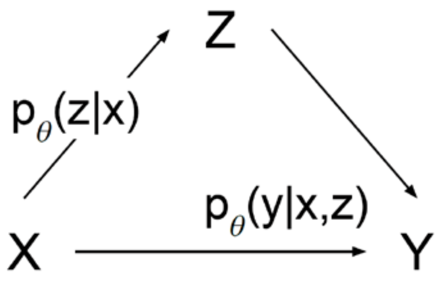


CVAE as CGM version:

$$\log P_\theta(y|x) \geq \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta(y|x,z)] - \boldsymbol{KL}(q_\phi(z|x,y)||p_\theta(z|x))$$

Theory background
CVAE as deep-CGM

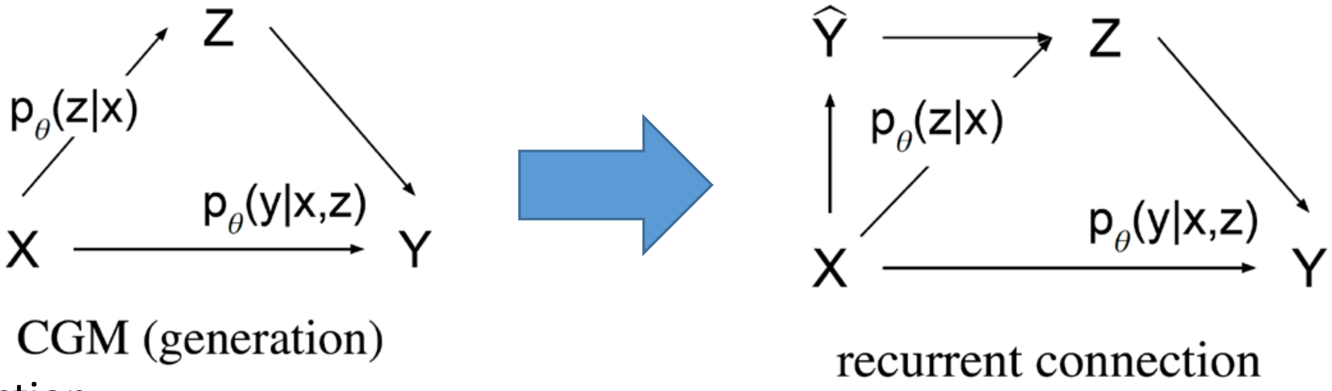
- Understanding CVAE as CGM (Conditional Generative Model)
- By doing this sampling of $P_\theta\ (y|x,z)$, $\mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta\ (y|x,z)]$ can be replaced.
- CVAE as CGM version:
- $\log P_\theta(y|x)\ \geq\ -\boldsymbol{KL}(q_\phi(z|x,y)|)||\,p_\theta(z|x)) + \mathbf{E}_{q_\phi(z|x,y)}[\log P_\theta\ (y|x,z)]$
- The <u>empirical lower bound</u> is written as:
- $\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(y|x,z^{(l)})$



CGM (generation)

Theory background
Recurrent CVAE

- Since we use CVAE as CGM all the time, we just call it CVAE.
- $\hat{Y}$, which is generated from a base line CNN, works as extra information for $p_\theta(z|x)$.
- So in fact $p_\theta(z|x) := p_\theta(z|x,\hat{y})$, but since $\hat{y}$ is generated by CNN only depends on x, we still use the notation $p_\theta(z|x)$.
- Significant <u>performance improvement</u> because of the extra info .

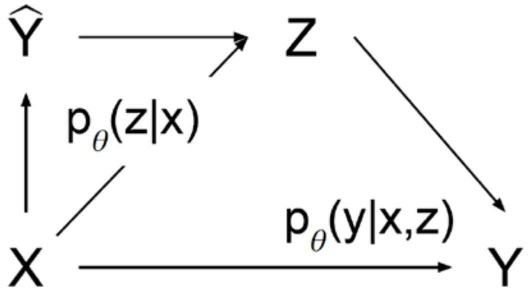


CVAE object function

$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(y|x,z^{(l)})$$

Theory background
Recurrent CVAE

- Let's summarize what we have now.
- We have a Recurrent CVAE model.
- We have several distributions:
- 1. Recognition model $q_\phi(z|x, y)$: to approximate real $p_\theta(z|x)$ distribution.
- 2. Prior model $p_\theta(z|x) := p_\theta(z|x, \hat{y})$: generate latent state from $x, \hat{y}$
- 3. Generation model $p_\theta(y|x, z)$: generate target output y
- 4. Base CNN: generate the initial guess $\hat{y}$ of y to sample prior
- Since we are doing a <u>deep generative model</u>, all the models mentioned above are deep CNNs.

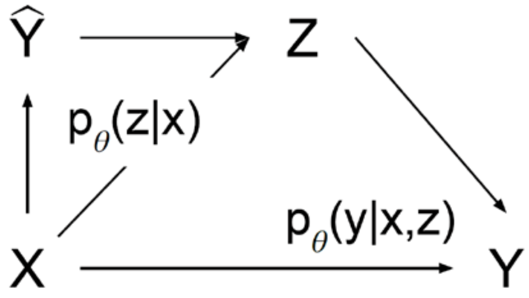


recurrent connection

CVAE object function

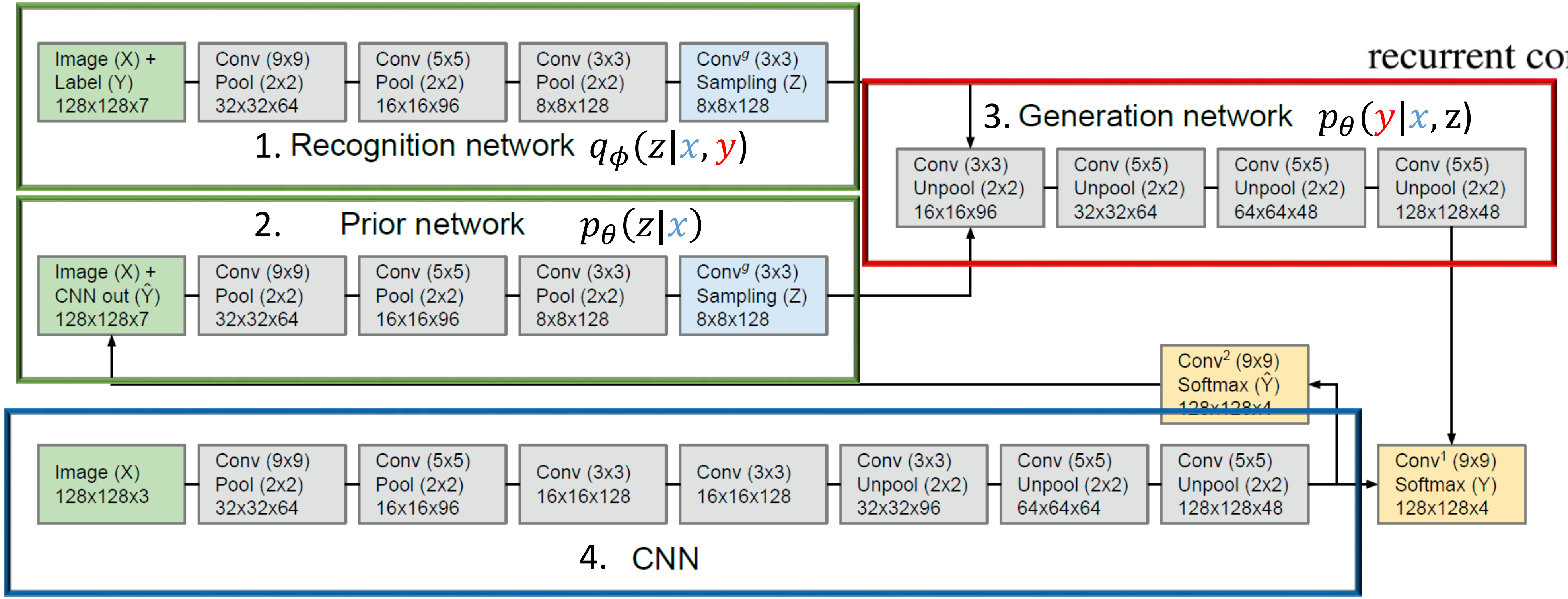
$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y) \| \, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log \; p_\theta(y|x, z^{(l)})$$

Model setup



- 4 Networks structures

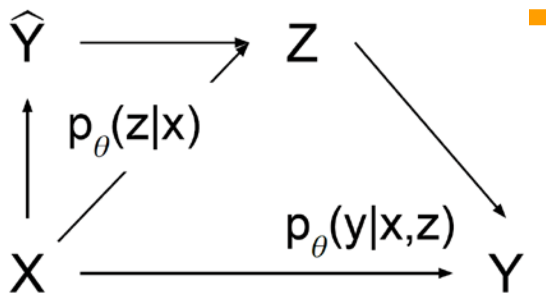
recurrent connection



CVAE object function

$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)\,||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x,z^{(l)})$$

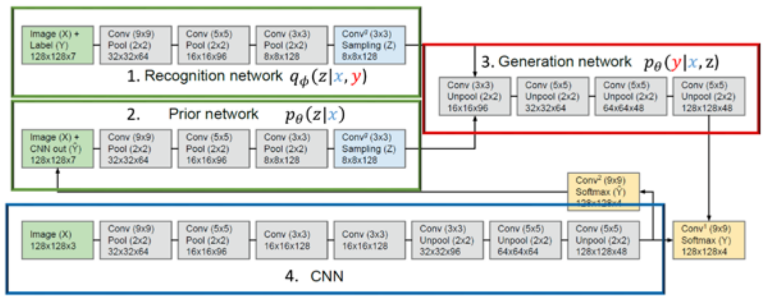
Model setup

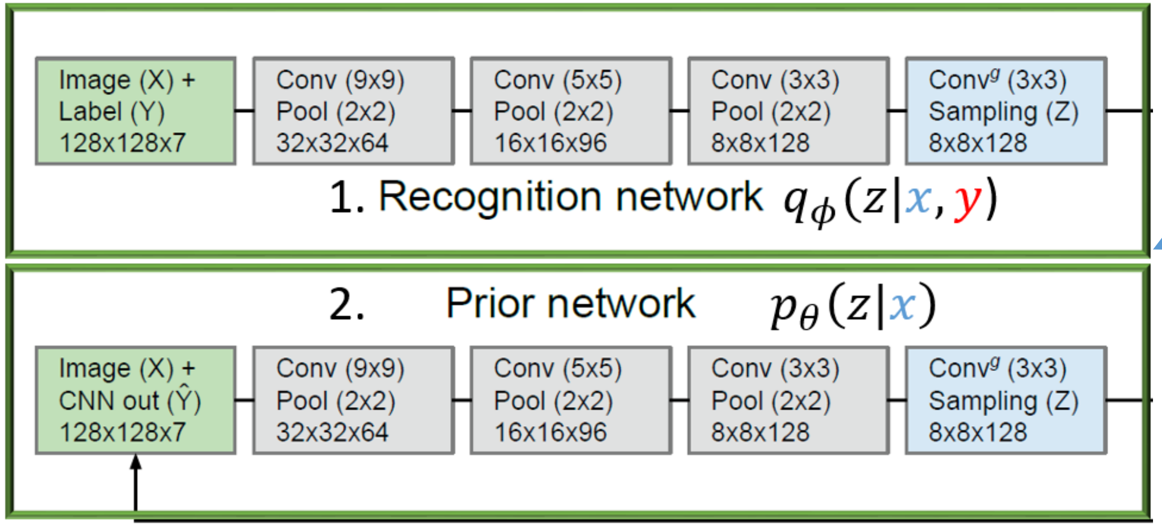


recurrent connection

- 1&2: Recognition network and Prior network
- They have the same structure.
- Input image dimension: x 128*128*3
- Output image dimension: y 128*128*4: extra channel for label
- Input size: concatenate x and y : 128*128*(4+3)
- Output sampled z: 8*8*128

- 4 Networks structures



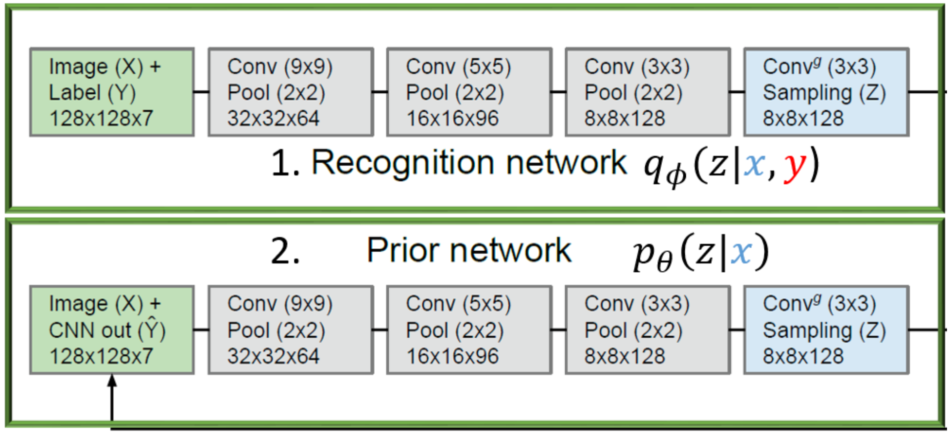


CVAE object function

$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(y|x,z^{(l)})$$

Model setup

- 1&2: Recognition network and Prior network



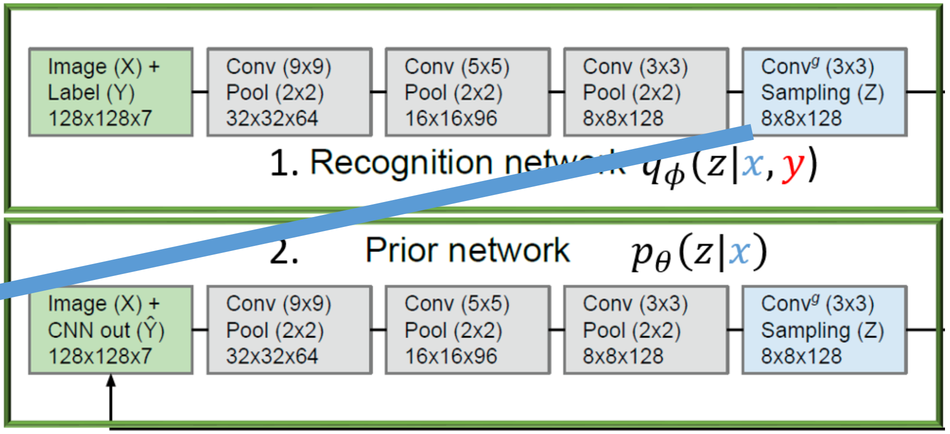
layer	op.	size-in	size-out	kernel
1	conv	128×128×7	64×64×64	9×9×7
	pool	64×64×64	32×32×64	2×2×1
	relu	32×32×64	32×32×64	–
2	conv	32×32×64	32×32×96	5×5×64
	pool	32×32×96	16×16×96	2×2×1
	relu	16×16×96	16×16×96	–
3	conv	16×16×96	16×16×128	3×3×96
	pool	16×16×128	8×8×128	2×2×1
	relu	8×8×128	8×8×128	–
4	conv^g	8×8×128	8×8×32	3×3×128
	sampling	8×8×32	8×8×32	–

Table S2: Prior and recognition networks definition. "conv^g" refers the layer that outputs Gaussian latent variables, and it includes convolution for mean and standard deviation units followed by gaussian sampling ("sampling").

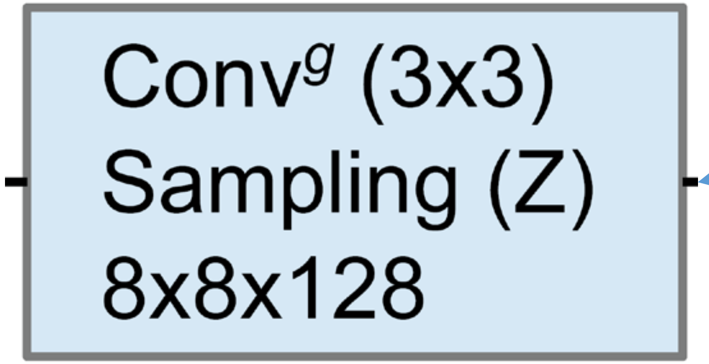
CVAE object function

$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y)\,||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x, z^{(l)})$$

Model setup



- 1&2: Recognition network and Prior network



- Apply Re-parametrization Trick here. $\mathbf{z}^{(l)} = g_\phi(\mathbf{x}, \mathbf{y}, \epsilon^{(l)}),\ \epsilon^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- The network works as the $g_\phi(.,.,.)$, deterministic, differentiable function, whose arguments are data x, y and the noise variable ε.
- This trick allows error backpropagation through the Gaussian latent variables, which is essential training as it is composed of multiple MLPs.
- As a result, the CVAE can be trained efficiently using stochastic gradient descent (SGD).

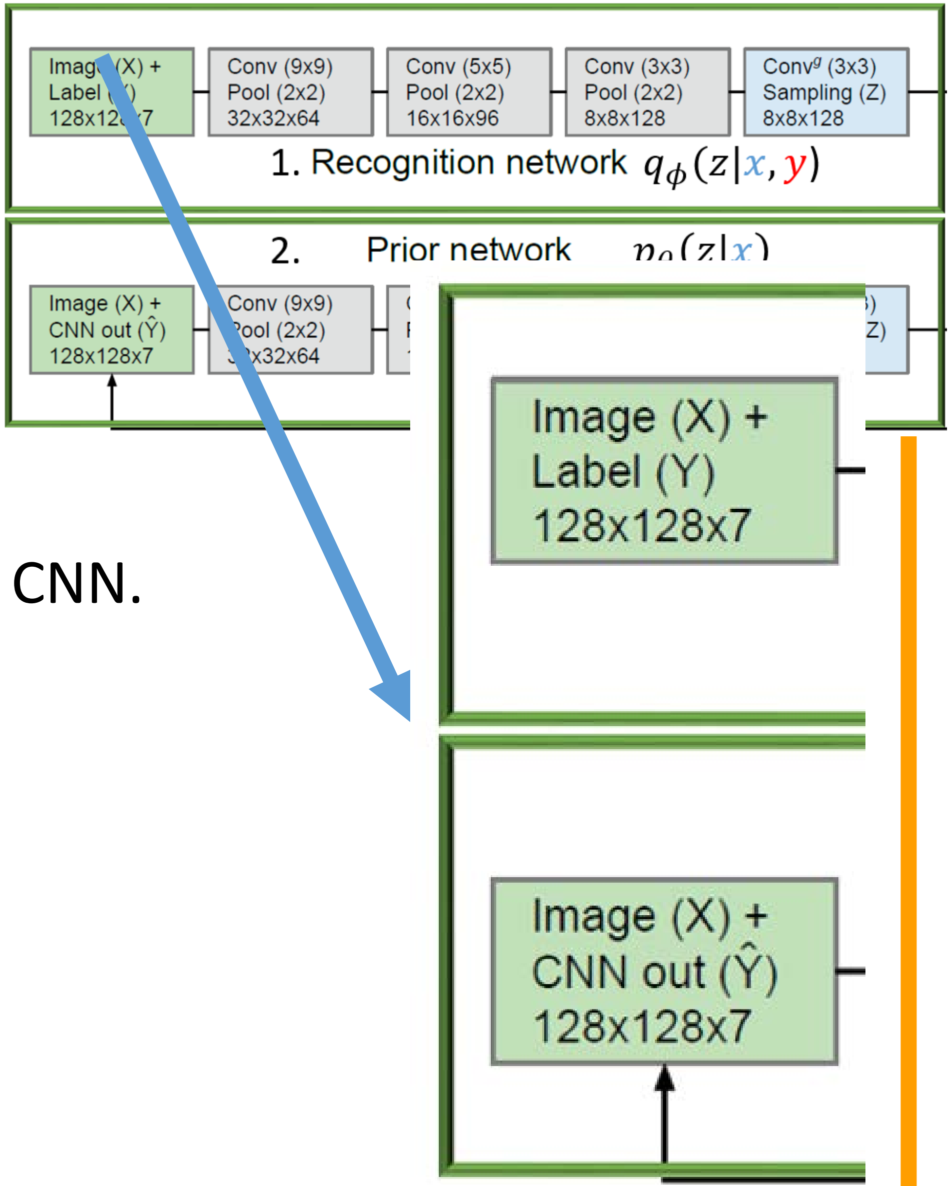
CVAE object function

$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x,y) \,||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x, z^{(l)})$$

Model setup

- 1&2: Recognition network and Prior network
- The difference between these two networks:

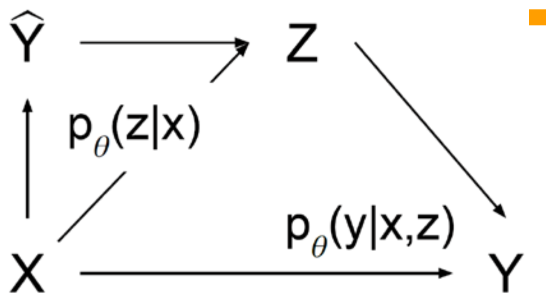
- Recognition network accepts x and real y.
- This network is <u>only used at training stage</u>, since it requires real label y.
- Prior network accepts x and estimator $\hat{y}$ generated by CNN.
- This network is <u>used both at training and prediction.</u>



CVAE object function

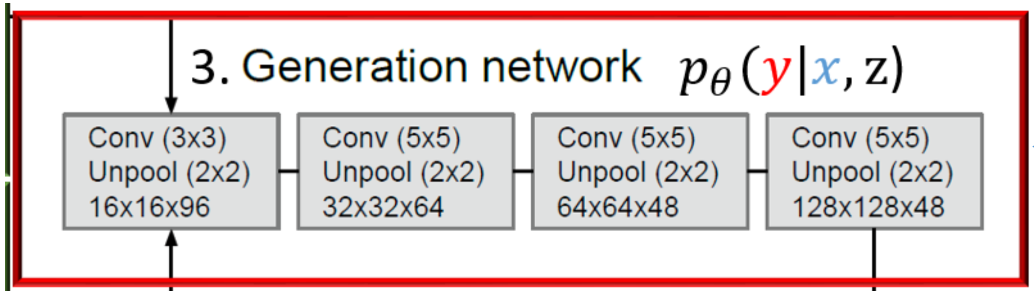
$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(y|x,z^{(l)})$$

Model setup

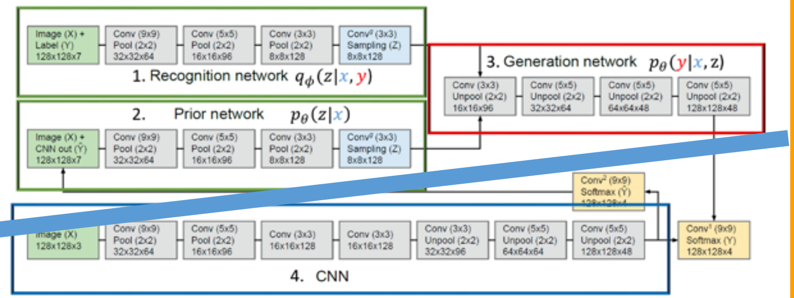


recurrent connection

- 3: Generation network
- Regular de-convolution network.
- Generate final output image 128*128*4 from 8*8*128 latent space.



- 4 Networks structures

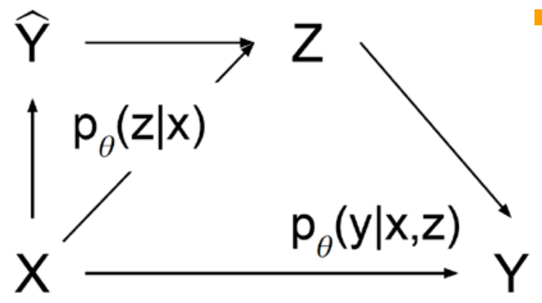


CVAE object function

CVAE object function

$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y) || \, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log \; p_\theta(y|x, z^{(l)})$$

Model setup



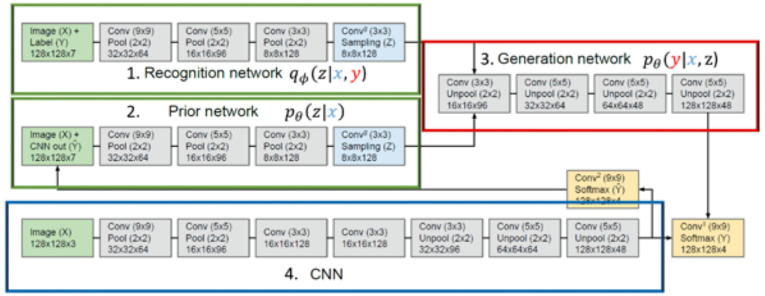
recurrent connection

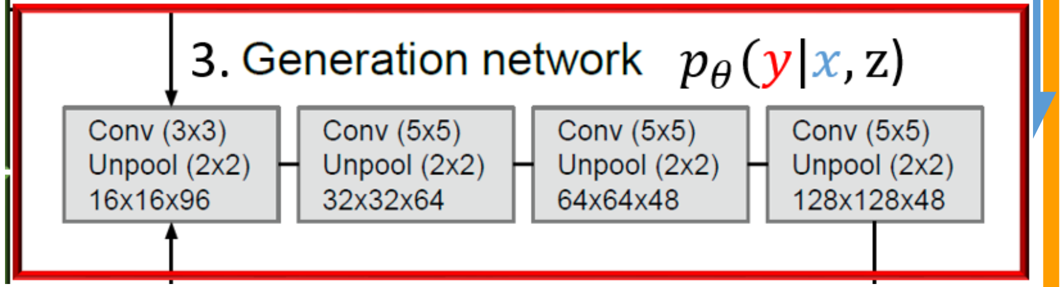
- 3: Generation network

layer	op.	size-in	size-out	kernel
1	conv	8×8×32	8×8×96	3×3×32
	unpool	8×8×96	16×16×96	2×2×1
	relu	16×16×96	16×16×96	–
2	conv	16×16×96	16×16×64	5×5×96
	unpool	16×16×64	32×32×64	2×2×1
	relu	32×32×64	32×32×64	–
3	conv	32×32×64	32×32×48	5×5×64
	unpool	32×32×48	64×64×48	2×2×1
	relu	64×64×48	64×64×48	–
4	conv	64×64×48	64×64×48	5×5×48
	unpool	64×64×48	128×128×48	2×2×1
	relu	128×128×48	128×128×48	–
5	conv	128×128×48	128×128×4	9×9×48
	softmax	128×128×4	128×128×4	–

Table S3: Generation network definition.

- 4 Networks structures

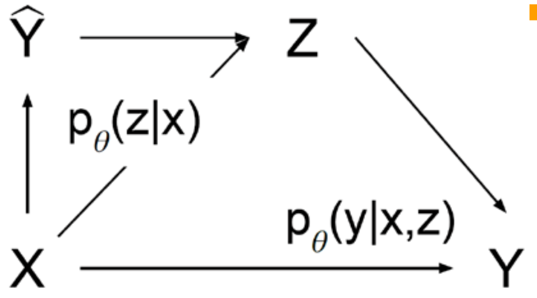




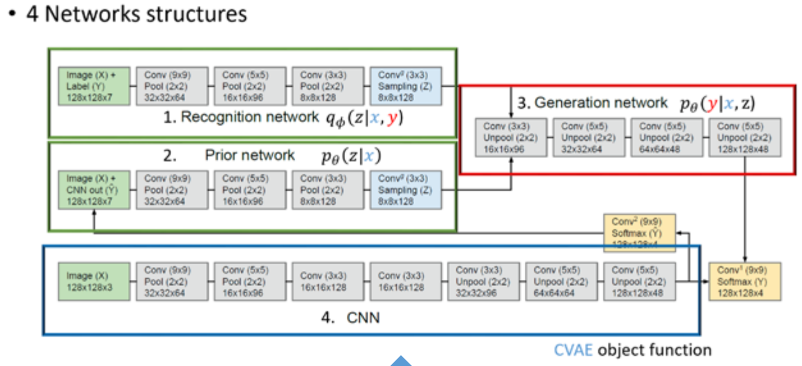
CVAE object function

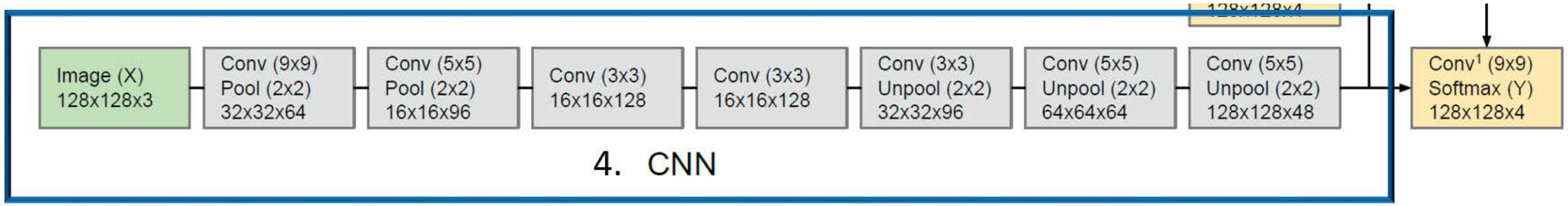
$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)\,||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(y|x,z^{(l)})$$

Model setup



- 4: Baseline CNN:
- Regular CNN structure
- Loss is defined as L2 distance between generated and $\hat{y}$ real output y.
- Used as the initial guess for prior network.

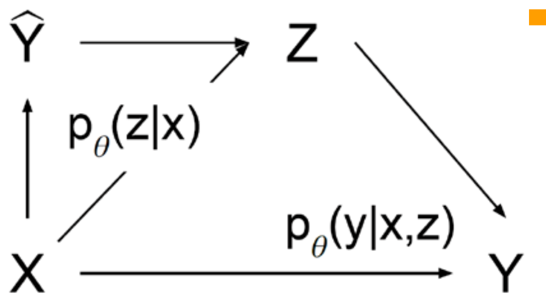




CVAE object function

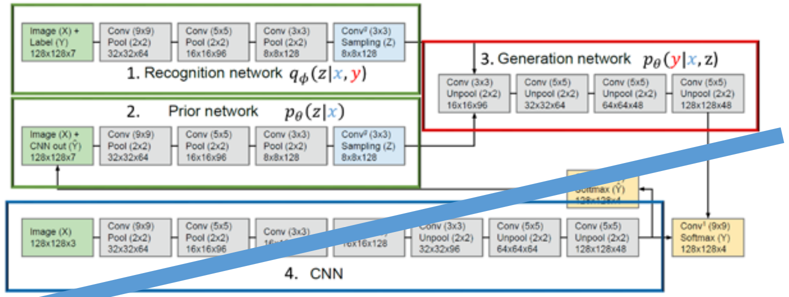
$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y)||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x, z^{(l)})$$

Model setup



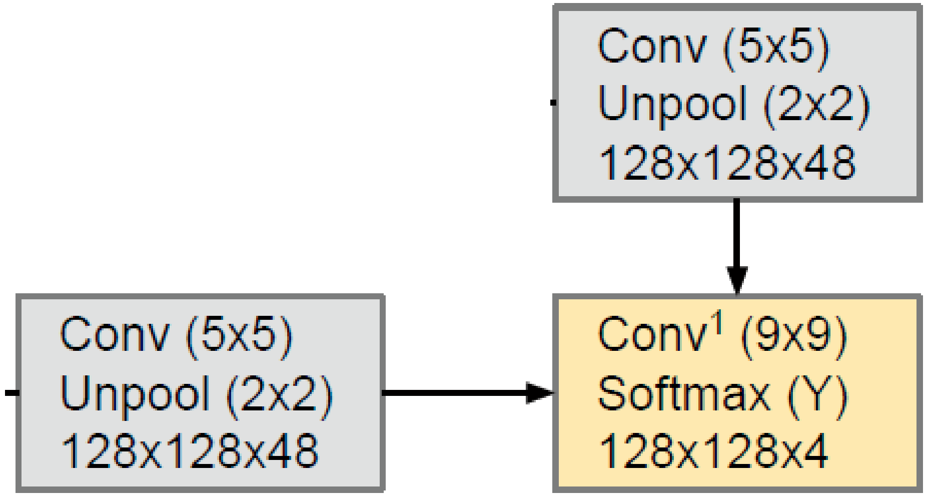
recurrent connection

- 4 Networks structures



CVAE object function

- Final output:
- The final output prediction is made by <u>element-wise summing the output of two convolutional networks</u>, which are the CNN and the generation network, followed by softmax classifier.



CVAE object function

$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)\,||\,p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L}\log\ p_\theta(y|x,z^{(l)})$$

Other Tricks
GSNN

- Problem:The CVAE uses the recognition network $q_\phi(z|x,y)$at training, but it uses the prior network $p_\theta(z|x)$ at testing to draw samples z's and make an output prediction.

- Since y is given as an input for the recognition network, the objective at training can be viewed as a reconstruction of y, which is an easier task than prediction.

- Proposed solution：
- Instead, we propose to train the networks in a way that the prediction pipelines at training and testing are consistent. This can be done by setting the recognition network the same as the prior network, i.e., $q_\phi(z|x,y) = p_\theta(z|x)$

CVAE object function

$$\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x,z^{(l)})$$

Other Tricks
GSNN

- Setting: $q_\phi(z|x,y) = p_\theta(z|x)$ means $\boldsymbol{KL}(q_\phi(z|x,y)||\, p_\theta(z|x)) = 0$
- Modify the target function:

- CVAE object function:
- $\tilde{L}_{VAE}(x,y;\phi,\theta) = -\boldsymbol{KL}(q_\phi(z|x,y)||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x,z^{(l)})$
- Gaussian stochastic neural network (GSNN) object function:
- $\tilde{L}_{GSNN}(x,y;\phi,\theta) = \cancel{-\boldsymbol{KL}(q_\phi(z|x,y)||\, p_\theta(z|x))} + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x,z^{(l)})$

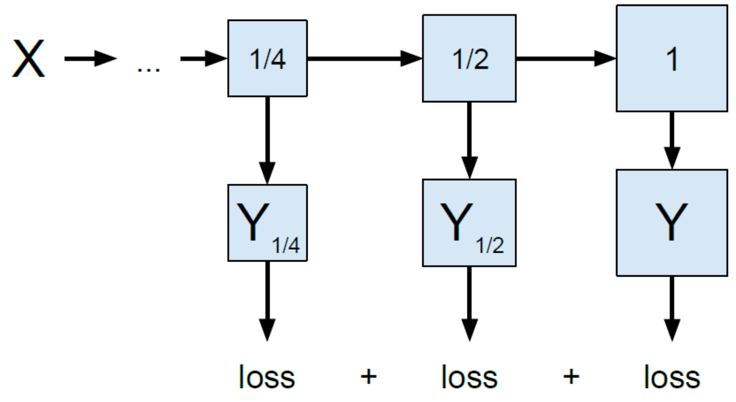
  where $\mathbf{z}^{(l)} = g_\theta(\mathbf{x}, \epsilon^{(l)}),\ \epsilon^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

- Further do a hybrid objective which combines GSNN and CVAE:
- $\tilde{L}_{hybird} = \alpha\, \tilde{L}_{VAE} + (1-\alpha)\tilde{L}_{GSNN}$
- Where α balances the two objectives.

Other Tricks
image processing

- Problem: To learn a high-capacity neural network that can be generalized well to unseen data, regularization is needed.
- 1. Training with multi-scale prediction objective

Here, we propose to train the network to predict outputs at different scales.



Right figure shows the multi-scale prediction at 3 different scales (1/4, 1/2, and original) for the training.

For testing, always use full size i.e. 128*128*3.

CVAE object function

$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y)||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x, z^{(l)})$$

Other Tricks
image processing

- Problem: To learn a high-capacity neural network that can be generalized well to unseen data, regularization is needed.
- 2. Training with input omission noise

Adding noise to neurons is a widely used technique to regularize deep neural networks during the training.

- corrupt the input data x into $\tilde{x}$ according to noise process and optimize the network with the following objective: $\tilde{L}(\tilde{x}, y)$

- Set no more than 40% width by 40% height random region = 0.

CVAE object function

$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_{\phi}(z|x, y)||\, p_{\theta}(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_{\theta}(y|x, z^{(l)})$$

Output Evaluation

- To evaluate the CGMs is to compare the conditional likelihoods of the test data. A straightforward approach is to draw samples z's using the prior network and take the average of the likelihoods. We call this method the Monte Carlo (MC) sampling:
- $p_\theta(y|x) \approx \frac{1}{S}\sum_{i=1}^{S} p_\theta\left(y \middle| x, z^{(s)}\right), z^{(s)} \sim p_\theta(z|x)$
- However, MC needs about 10, 000 samples per example to get an accurate estimate.

- We can improve it by introducing importance sampling:
- $p_\theta(y|x) \approx \frac{1}{S}\sum_{i=1}^{S} \frac{p_\theta\left(y \middle| x, z^{(s)}\right) p_\theta\left(z^{(s)}|x\right)}{q_\phi\left(z^{(s)}|x, y\right)}, z^{(s)} \sim q_\phi(z|x, y)$
- By doing so, 100 samples are good for estimate.

CVAE object function

$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y) || \, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log \; p_\theta(y|x, z^{(l)})$$

Test results

- 1. Visual Object Segmentation and Labeling
- Caltech-UCSD Birds (CUB): 6, 033 images of birds from 200 species birds

*GDNN: If we assume a covariance matrix of auxiliary Gaussian latent variables ε to 0, we have a deterministic counterpart of GSNN, which we call a Gaussian deterministic neural network (GDNN).
* "msc" refers multi-scale prediction training and "NI" refers the noise-injection training

In terms of prediction accuracy, the GSNN performed the best among our proposed models. It is designed to predict well.

Model (training)	CUB (val)		CUB (test)	
	pixel	IoU	pixel	IoU
MMBM [37]	–	–	90.42	75.92
GLOC [13]	–	–	–	–
CNN (baseline)	91.17 ±0.09	79.64 ±0.24	92.30	81.90
CNN (msc)	91.37 ±0.09	80.09 ±0.25	92.52	82.43
GDNN (msc)	92.25 ±0.09	81.89 ±0.21	93.24	83.96
GSNN (msc)	92.46 ±0.07	82.31 ±0.19	93.39	84.26
CVAE (msc)	92.24 ±0.09	81.86 ±0.23	93.03	83.53
hybrid (msc)	92.60 ±0.08	82.57 ±0.26	93.35	84.16
GDNN (msc, NI)	92.92 ±0.07	83.20 ±0.19	93.78	85.07
GSNN (msc, NI)	**93.09** ±0.09	**83.62** ±0.21	**93.91**	**85.39**
CVAE (msc, NI)	92.72 ±0.08	82.90 ±0.22	93.48	84.47
hybrid (msc, NI)	**93.05** ±0.07	**83.49** ±0.19	**93.78**	**85.07**

CVAE object function: $\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y) || \, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log \; p_\theta(y|x, z^{(l)})$

Test results

- 2. Object Segmentation with Partial Observations
- We randomly omit the input pixels at different levels of omission noise (25%, 50%, 70%) and different block sizes (1, 4, 8), and the task is to predict the output segmentation labels(4th channel) for the omitted pixel locations while given the partial labels for the observed input pixels.
- To make a prediction for CVAE with partial output observation (y_o), we perform iterative inference of unobserved output (y_u) and the latent variables (z)
- $y_u \sim p_\theta(y_u|x, z) \leftrightarrow z \sim q_\phi(z|x, y_o, y_u)$

CVAE object function: $\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y)||\, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x, z^{(l)})$

Test results

- 2. Object Segmentation with Partial Observations

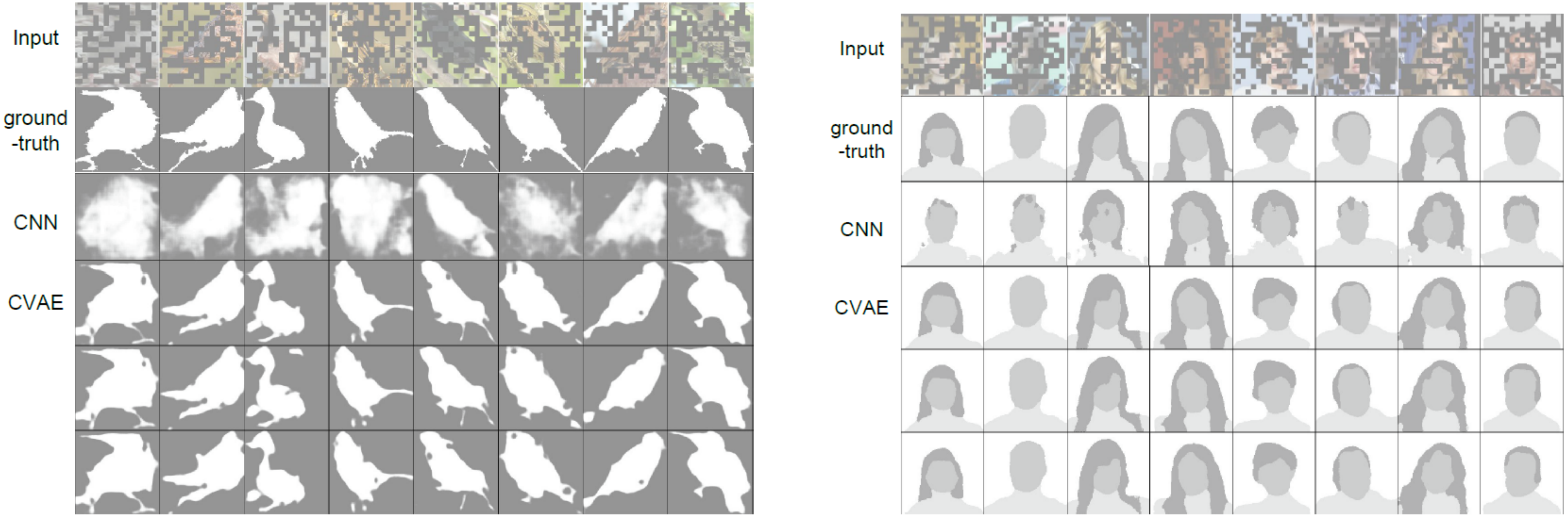


Figure 4: Visualization of the conditionally generated samples: (first row) input image with omission noise (noise level: 50%, block size: 8), (second row) ground truth segmentation, (third) prediction by GDNN, and (fourth to sixth) the generated samples by CVAE on CUB (left) and LFW (right).

CVAE object function: $\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y)||\ p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log\ p_\theta(y|x, z^{(l)})$

Test results

- 1. Object Segmentation with Partial Observations

The CVAE performs well even when the noise level is high (e.g., 50%), where the GDNN significantly fails.

This is because the CVAE utilizes the partial segmentation information to iteratively refine the prediction of the rest.

Dataset		CUB (IoU)		LFW (pixel)	
noise level	block size	GDNN	CVAE	GDNN	CVAE
25%	1	89.37	**98.52**	96.93	**99.22**
	4	88.74	**98.07**	96.55	**99.09**
	8	90.72	**96.78**	97.14	**98.73**
50%	1	74.95	**95.95**	91.84	**97.29**
	4	70.48	**94.25**	90.87	**97.08**
	8	76.07	**89.10**	92.68	**96.15**
70%	1	62.11	**89.44**	85.27	**89.71**
	4	57.68	**84.36**	85.70	**93.16**
	8	63.59	**76.87**	87.83	**92.06**

Table 4: Segmentation results with omission noise on CUB and LFW database. We report the pixel-level accuracy on the first validation set.

CVAE object function: $\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y) || p_\theta(z|x)) + \frac{1}{L}\sum_{l=1}^{L} \log\ p_\theta(y|x, z^{(l)})$

Conclusion

- Summary:

CVAE object function:

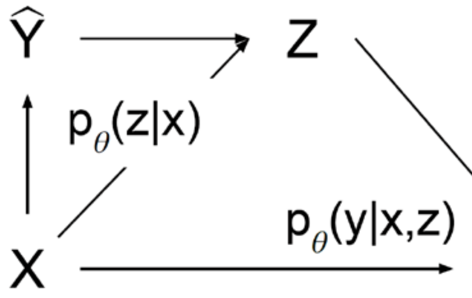
$$\tilde{L}_{VAE}(x, y; \phi, \theta) = -\boldsymbol{KL}(q_\phi(z|x, y) || \, p_\theta(z|x)) + \frac{1}{L}\sum_{i=1}^{L} \log \; p_\theta\left(y|x, z^{(l)}\right)$$

CSNN object function:

$$\tilde{L}_{GSNN}(x, y; \phi, \theta) = \frac{1}{L}\sum_{i=1}^{L} \log \; p_\theta(y|x, z^{(l)})$$

hybrid objective which combines GSNN and CVAE:

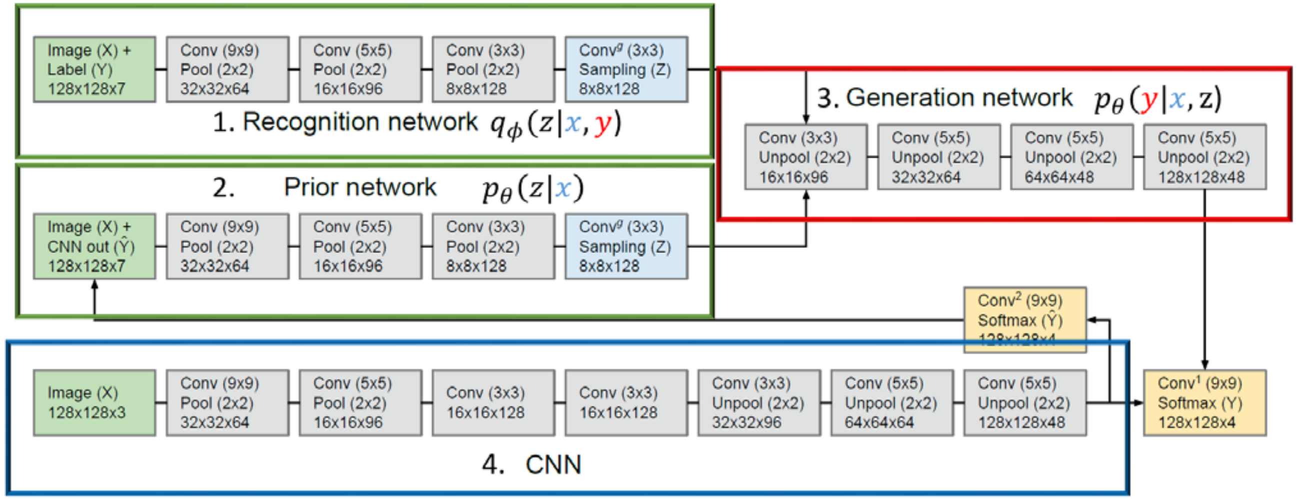
$$\tilde{L}_{hybird} = \; \alpha \, \tilde{L}_{VAE} + (1 - \alpha)\tilde{L}_{GSNN}$$



recurrent connection

Four networks:

1. Recognition model $q_\phi(z|x, y)$: to approximate real $p_\theta(z|x)$ distribution.
2. Prior model $p_\theta(z|x) := p_\theta(z|x, \hat{y})$: generate latent state from $x, \hat{y}$
3. Generation model $p_\theta(y|x, z)$: generate target output y
4. Base CNN: generate the initial guess $\hat{y}$ of y to sample prior



CVAE object function

Learning to Generate Chairs with Convolutional Neural Network

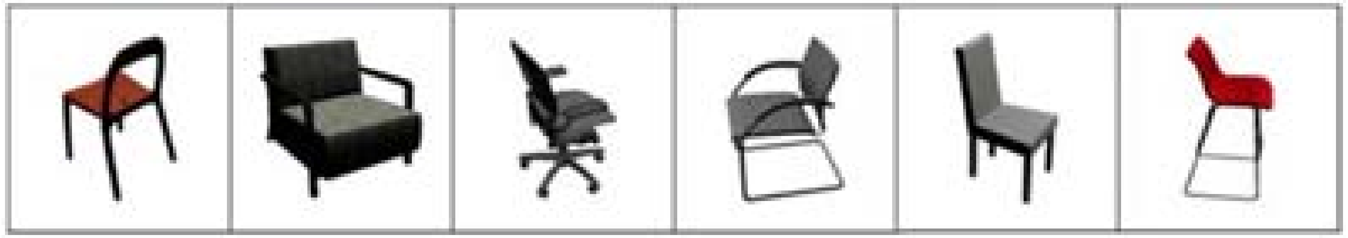
By Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, Thomas Brox

Generative CNN

- Previous Work
  - Graphical Model: CRF or MRF
  - Graphical Model + Deep network
- Major Difference
  - Using supervised learning and assuming high level representation is given
  - No inference procedure. Purely based on deep learning
- Discriminative CNN vs Generative CNN
  - Discriminative CNN: Image -> Viewpoint/Class/Style...
  - Generative CNN: Viewpoint/Class/Style.. ->Image

Model Description

- Generate an object based on high-level inputs
  - Class **(C)**
  - Viewpoint: Orientation with respect to camera **(V)**
  - Artificial Transformation (**Θ**)
    - Rotation, translation, zoom
    - Stretching horizontally or vertically
    - saturation, brightness, Hue



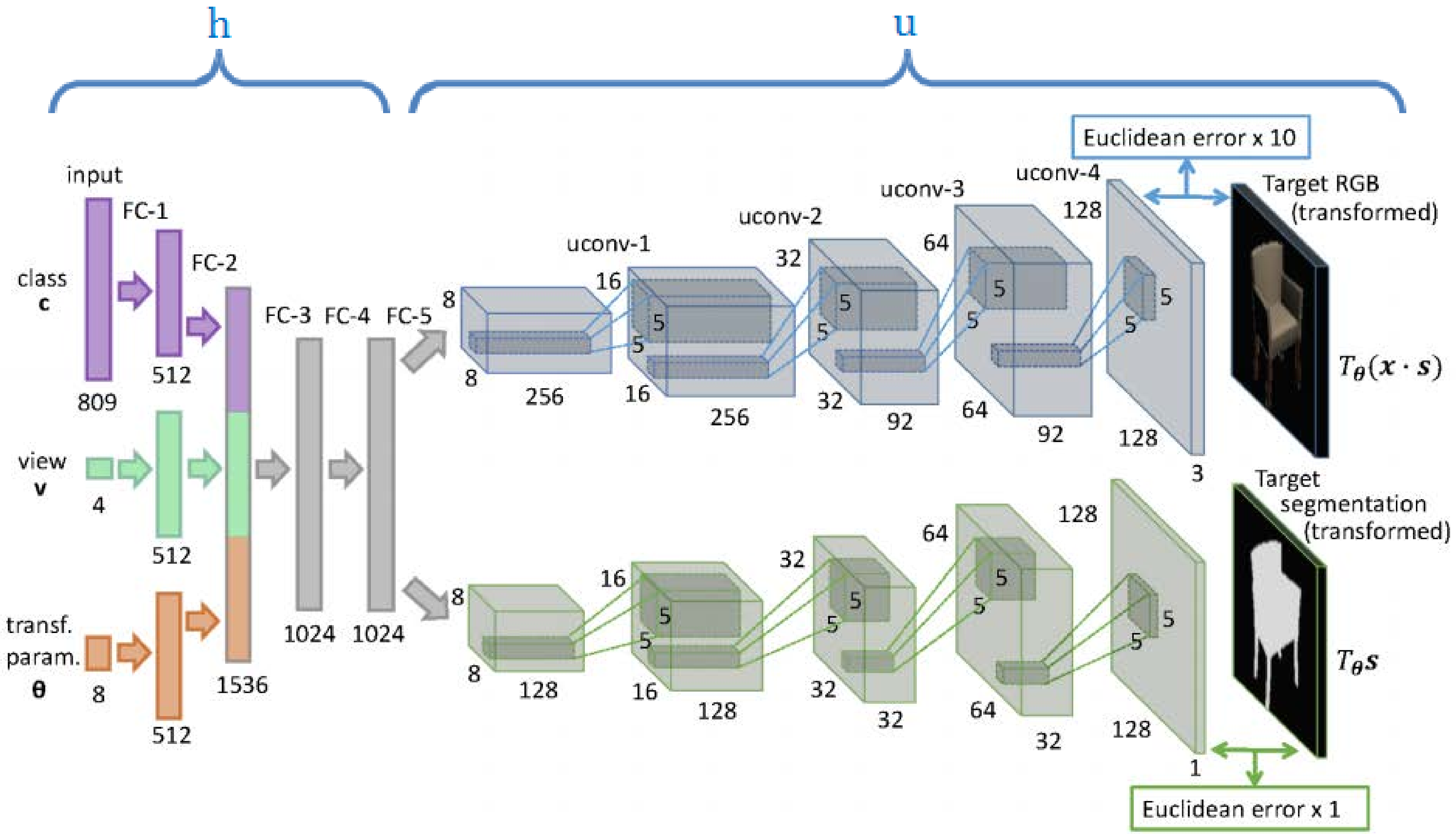
Dataset

- Using 3D chair model dataset
  - Original dataset: 1393 chair models, 62 viewpoints, 31 azimuth angles, 2 elevation angles
  - After Preprocessing: 809 models, tight cropping, resizing to 128x128/64*64
- Input and Output Notation

$$D = \{(\mathbf{c}^1, \mathbf{v}^1, \theta^1), \ldots, (\mathbf{c}^N, \mathbf{v}^N, \theta^N)\}$$

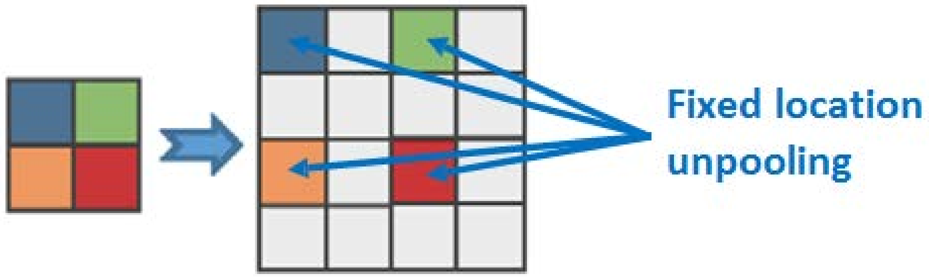
$$O = \{(\mathbf{x}^1, \mathbf{s}^1), \ldots, (\mathbf{x}^N, \mathbf{s}^N)\}$$

Architecture

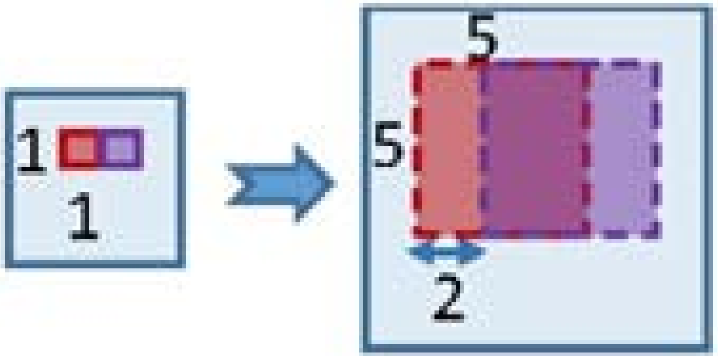


Operation

- Unpooling: 2x2



- Unpooling + Convolution



Training

- Objective Function:
  - Optimize the reconstruction Euclidean Loss

$$\min_{\mathbf{W}} \sum_{i=1}^{N} \lambda \| u_{RGB}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i)) - T_{\theta^i}(\mathbf{x}^i \cdot \mathbf{s}^i) \|_2^2$$
$$+ \| u_{segm}(h(\mathbf{c}^i, \mathbf{v}^i, \theta^i)) - T_{\theta^i}\mathbf{s}^i \|_2^2,$$

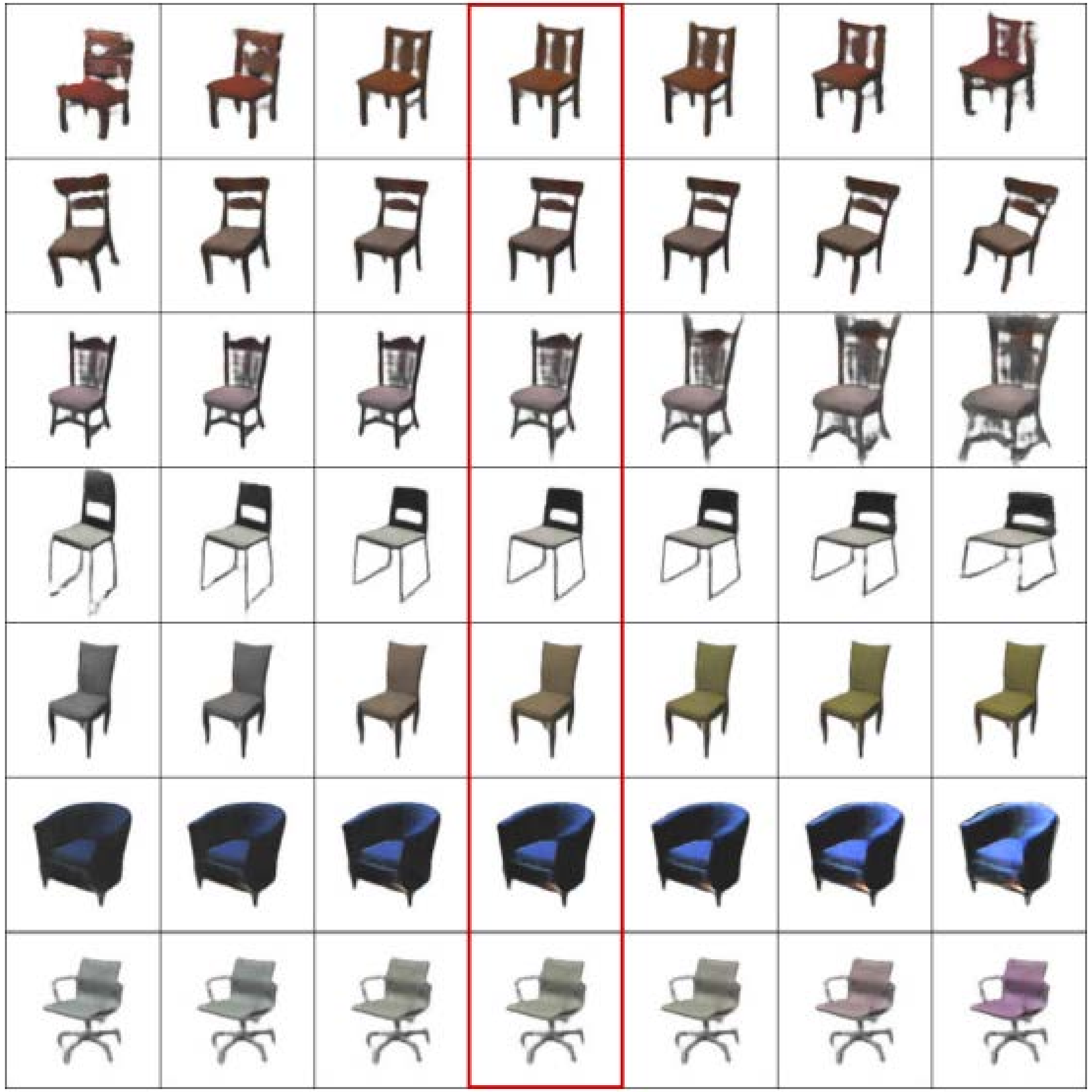
- Optimization:
  - Stochastic gradient descent with momentum of 0.9
  - Learning rate
    - 0.0002 for the first 500 epochs
    - Dividing by 2 after every 100 epoch

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta).$$

$$\theta = \theta - v_t.$$

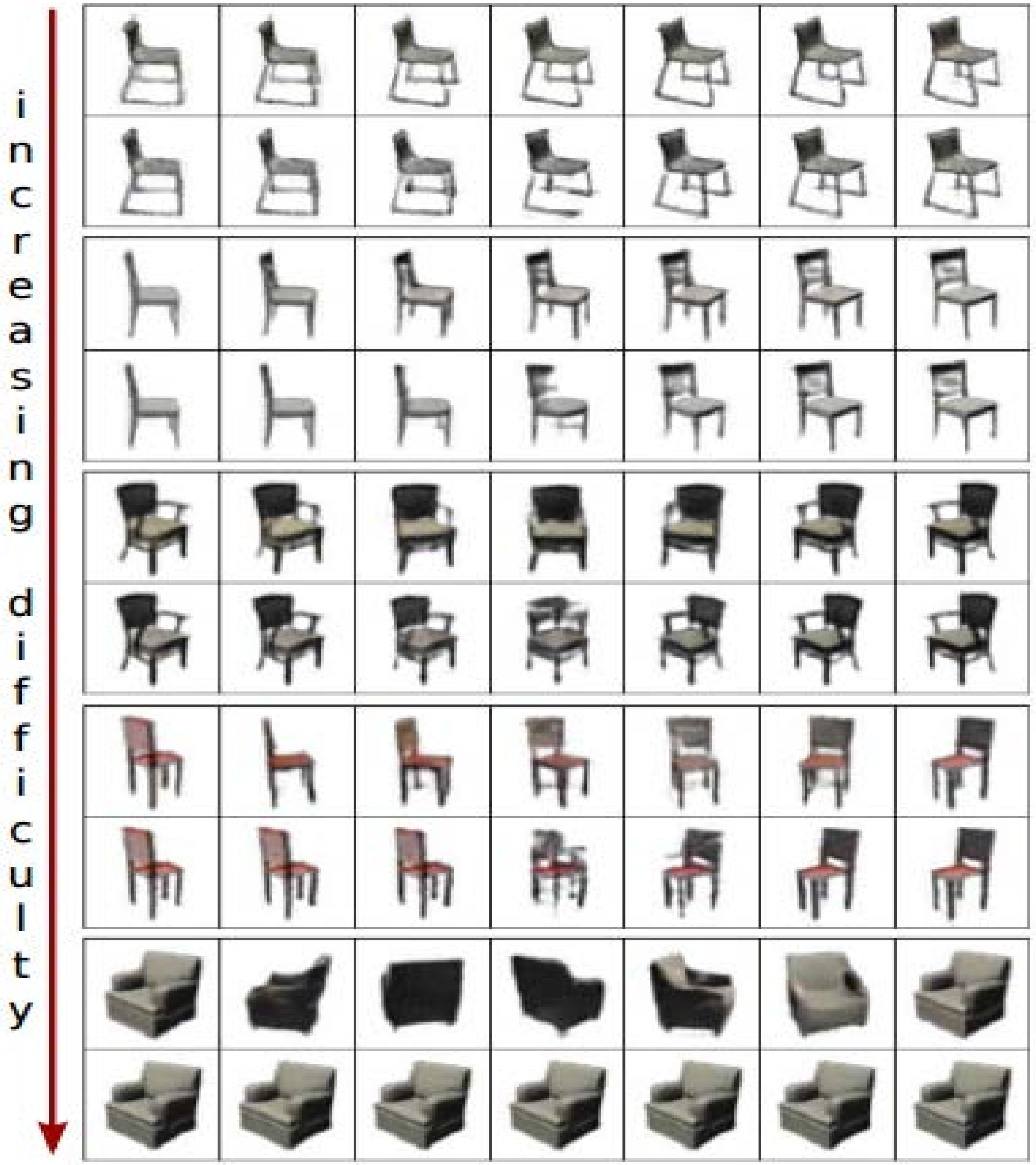
Network Capacity

- The network successfully Model the variation in the data
- Total number of parameter In the network: 32M Total number of pixel in training data>>400M
- Memorizing all the examples by heart is not an option



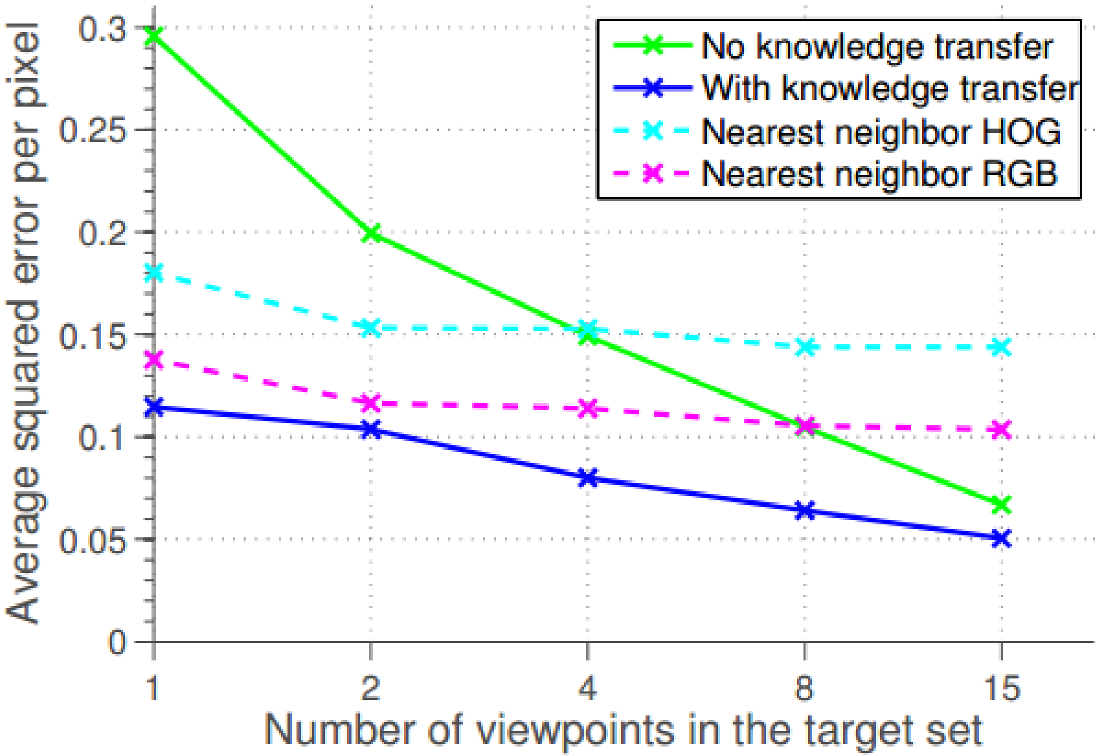
Knowledge Transfer

- Interpolation between Viewpoint
  - Left most image and right most image are provided during training
  - From top to bottom views use during training: 15, 8 ,4 ,2, 1
  - In each pair
    - Top row has knowledge transfer
    - Bottom row no knowledge transfer



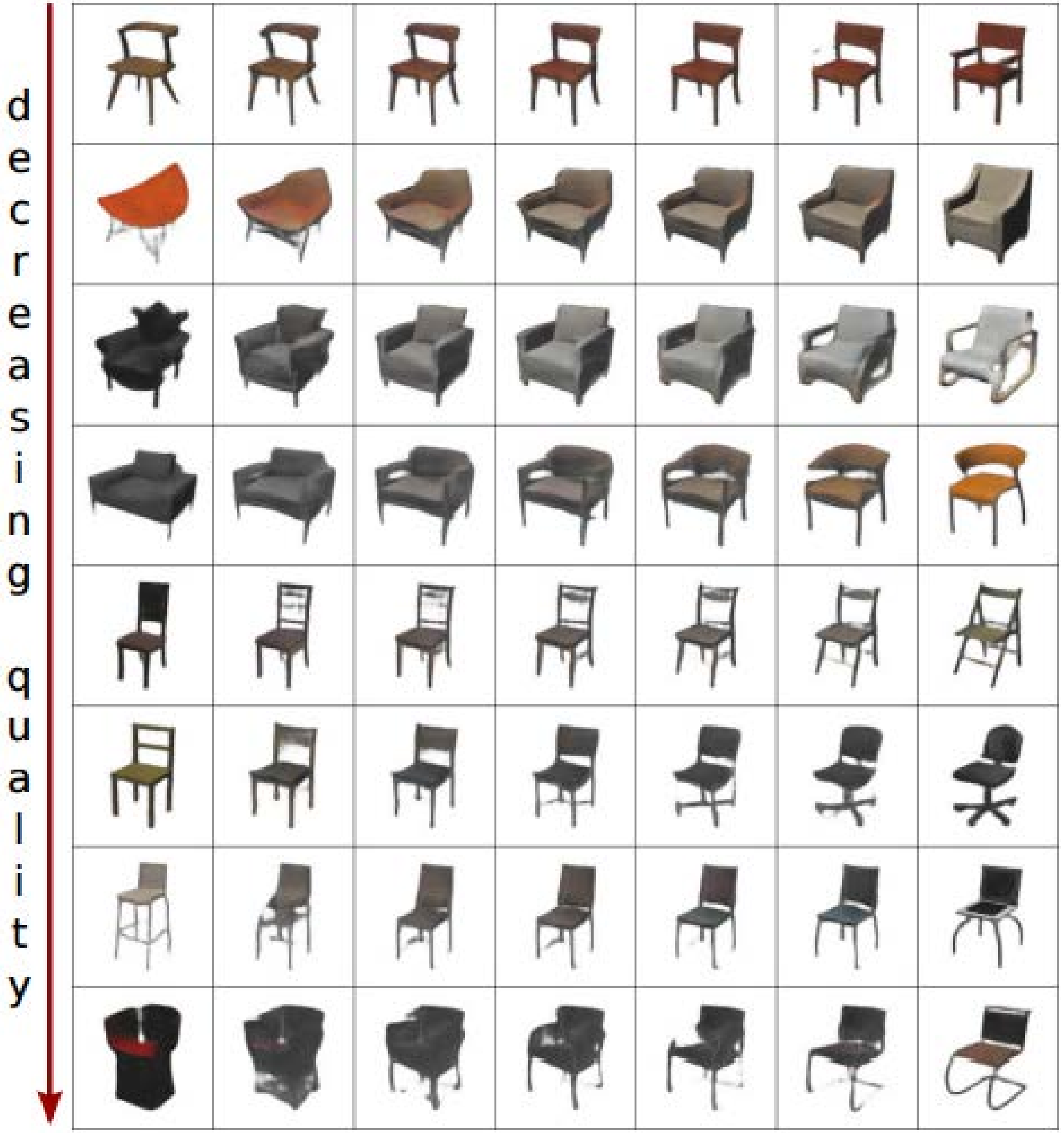
Knowledge Transfer

- How about represent missing views by nearest neighbors?



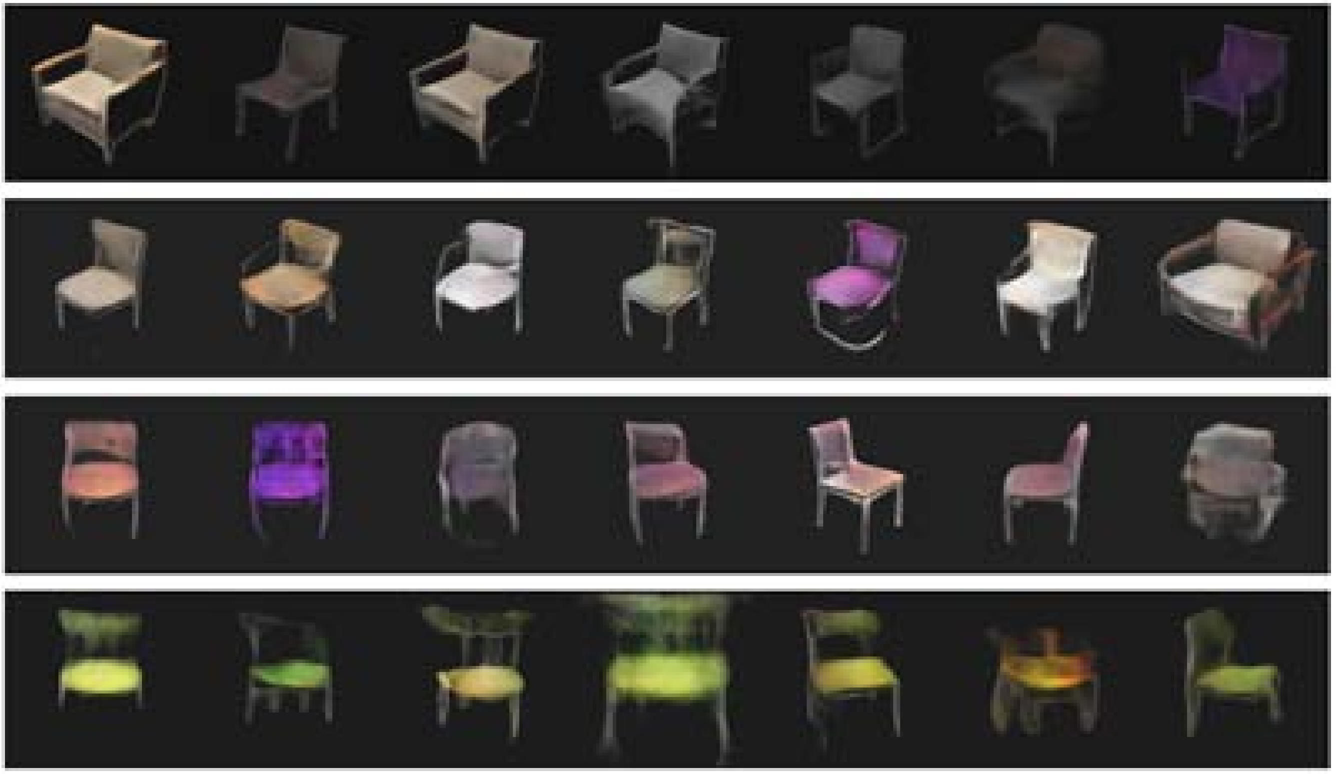
Style Interpolation

- Intermediate Styles:
  - Linearly change the input label vector from one class to another
  - Only given the first column and last column of the chairs, all the chair style in between is interpolated style by The network



Single Unit Activation

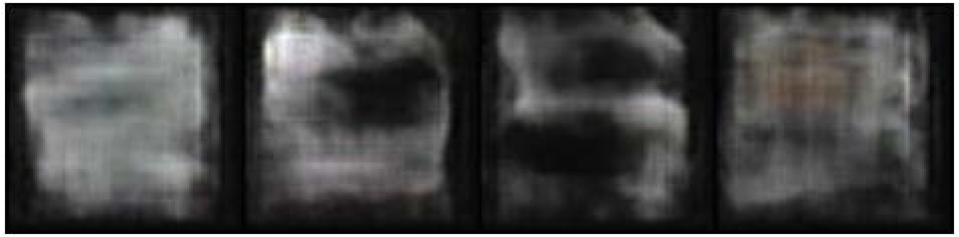
- Images generated from single neuron activations
  - From top to bottom: FC-1, FC-2, FC-3, FC-4



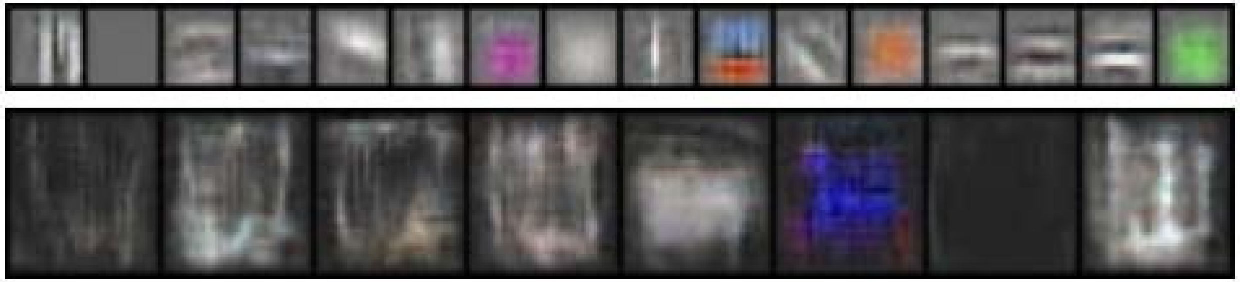
- Zoom Neuron

Single Unit Activation

- Image generated from single neuron activation in FC-5 layer

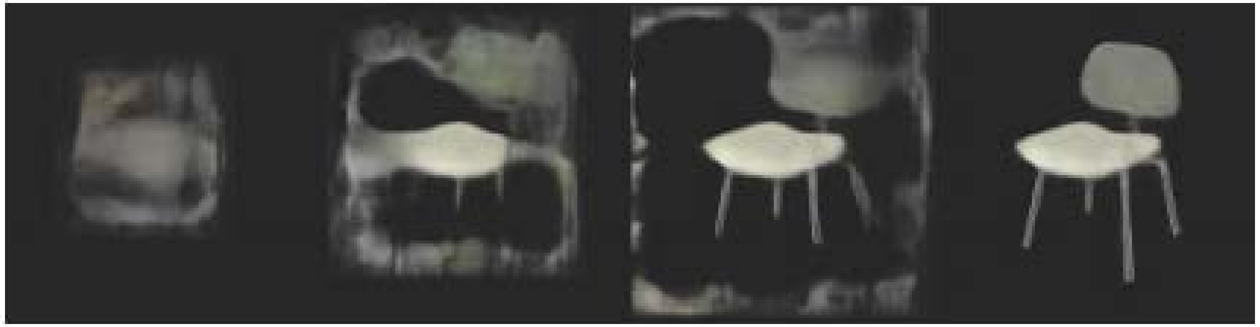


- Image generated from single neuron activation in higher deconvolutional layer
  - Top： uconv2 Bottom: uconv1



  - uconv1 is blur because of regular-grid unpooling
  - uconv2 produce edge-like images

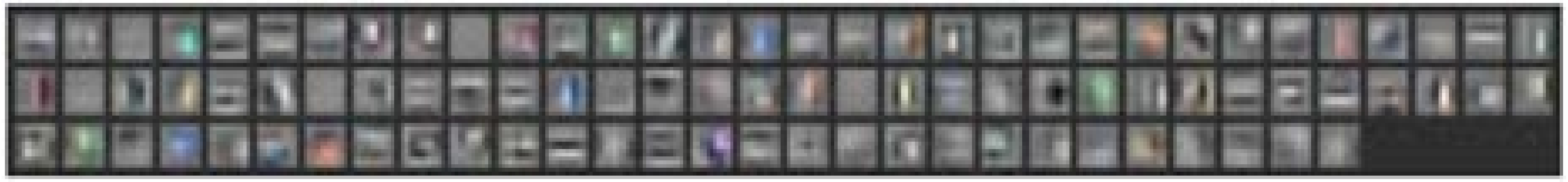
- Image generated by interpolate between single activation and whole chair



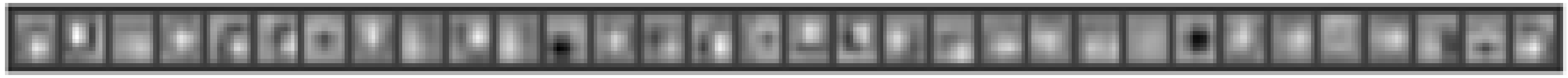
- Procedure
  - Gradually increase the size of neuron activation area around center of FC-5 feature maps
- Observation
  - Central region: sharp image
  - Peripheral region: blur image
  - Neighborhood neurons contributions

Filter Visualization

- Visualization of output layer filters in 128x128 network
  - RGB Stream

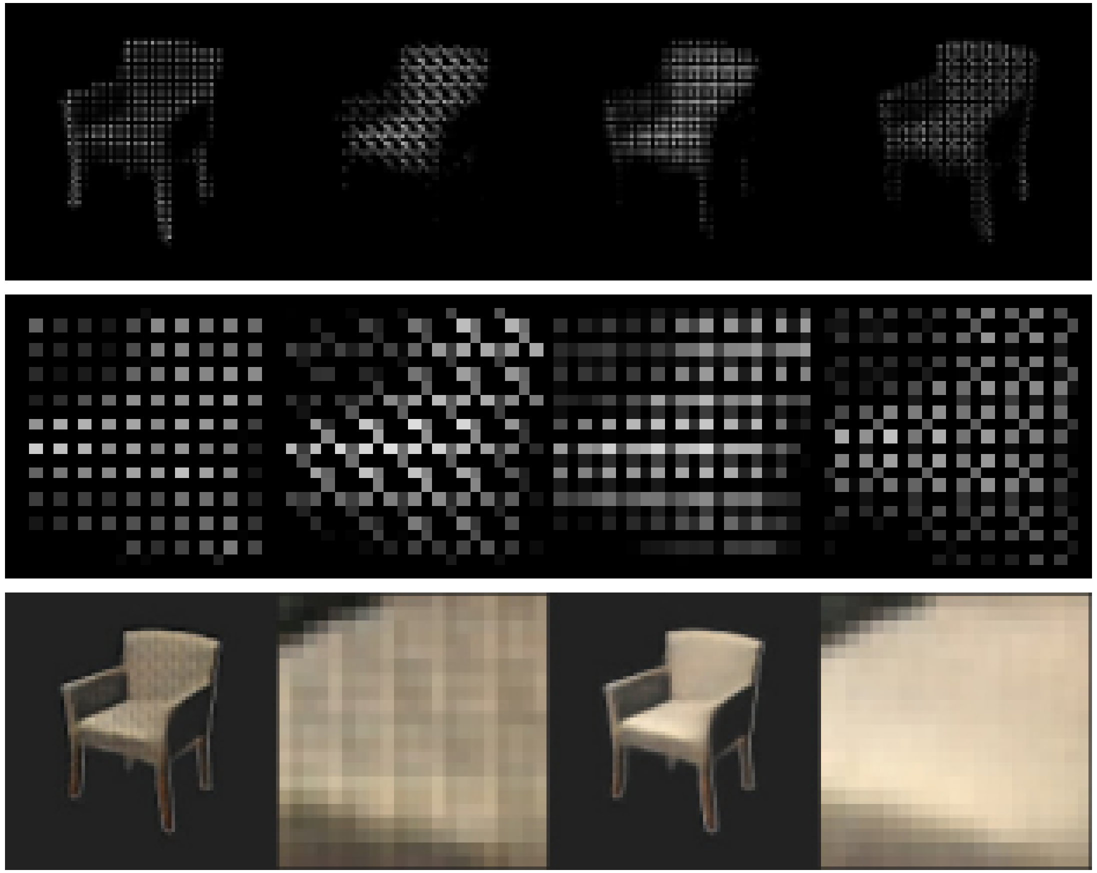


  - Segmentation



- Observation
  - The final output at each position is generated from a linear combination of these filters.
  - They include edges and blobs.

- Feature maps of uconv-3 and final images generated
  - Top: feature maps of uconv-3
  - Middle: close-ups of the feature maps
  - Bottom: generations of chairs with feature maps setting to zero and leaving unchanged



- Problem

  Given two chairs, and points in a chair, find the corresponding points on the other chair

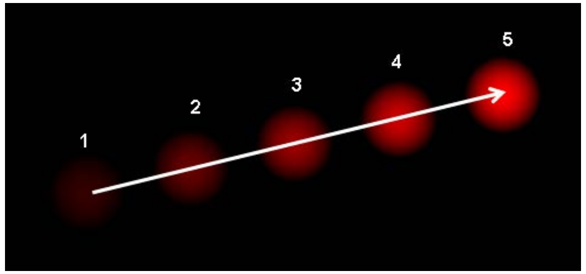
- Solution
  - Use the trained network to generate a morphing from the first chair to the second chair consisting of 64 images.
  - Compute refined optical flow for the generated image sequence
  - Concatenation of the optical flows gives the global vector connecting corresponding points of the two chairs.

- Morphing with fixed view

Optical flow

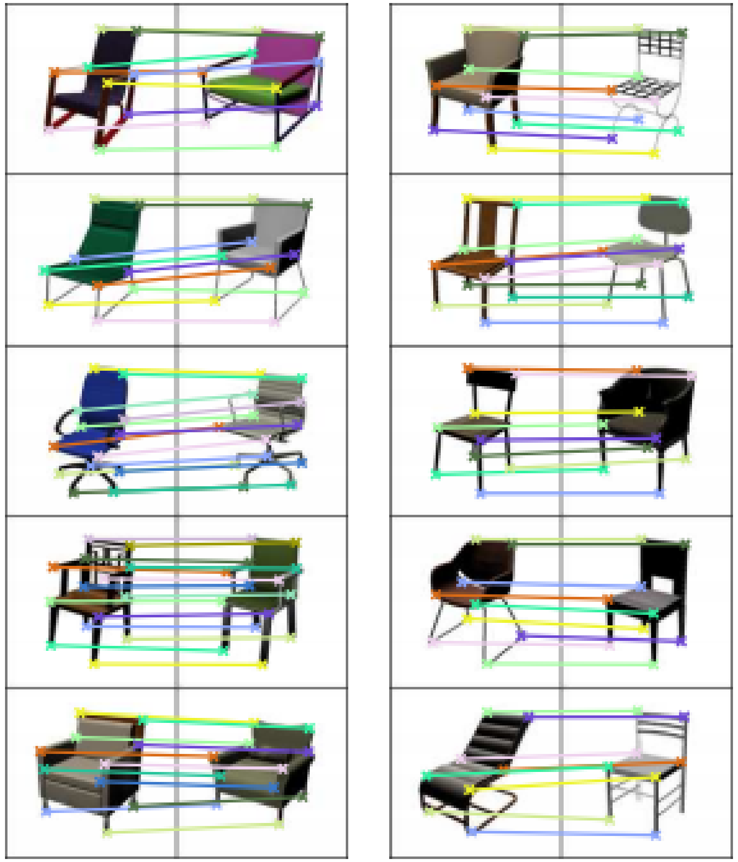
- Definition
  - Pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera.
  - It is 2D vector field where each vector is a displacement vector showing the movement of points from first frame to second.

- Example
  - An concatenation of optical flows for a moving ball in 5 consecutive frames



- Evaluation
  - Ground truth (show in left figure)
    - 9 people manually marked corresponding points and then use average position
  - Test
    - Use average displacement between predicted points and ground truth
  - Result and comparison (in pixel)

Method	All	Simple	Difficult
DSP [36]	5.2	3.3	6.3
SIFT flow [35]	4.0	**2.8**	4.8
Ours	**3.4**	3.1	**3.5**
Human	1.1	1.1	1.1



Conclusion

- Supervised training of CNN can be used for generating images
- Network indeed learns the implicit representation
- Can handle very different inputs – class labels, viewpoint and spatial information

THANKS!

Any questions?