Variational Autoencoders with Application to Unsupervised Representation Learning

Guo-Jun Qi, Liheng Zhang, Chang Wen Chen, Qi Tian; The IEEE International Conference on Computer Vision (ICCV), 2019

Yash Patel
Visual Recognition Group, Czech Technical University in Prague

Presentation Outline

1. Fundamentals of Variational Autoencoders.

2. Transformation Equivariant Representations.

3. AVT: Unsupervised Learning of Transformation Equivariant Representations by Autoencoding Variational Transformations (*Qi et al. ICCV 2019*).

Variational Autoencoders - Objective

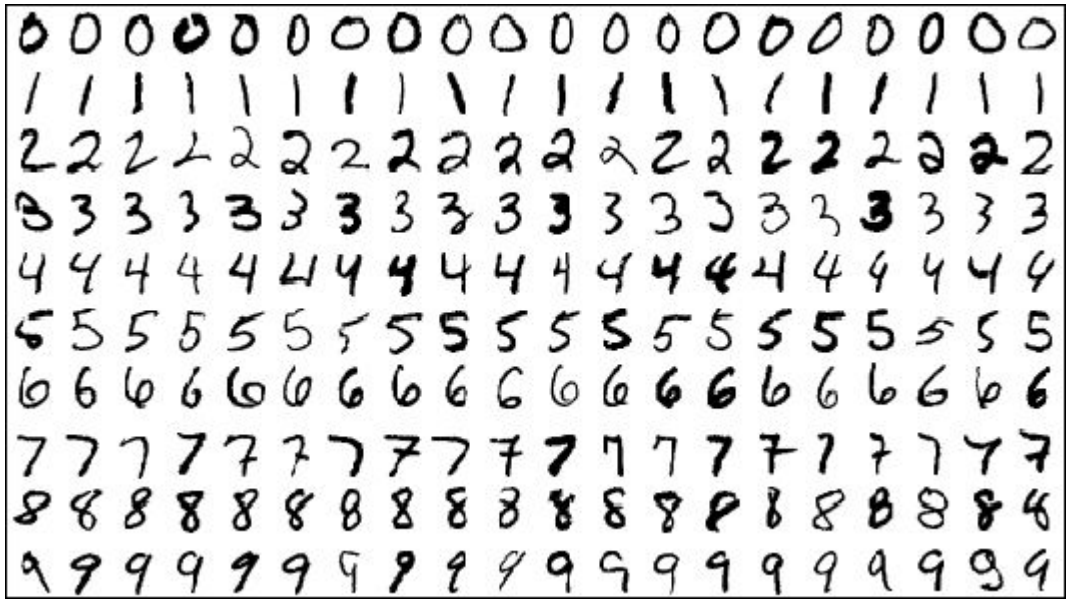
Unknown GT distribution

$$P_{\mathrm{GT}}$$

Objective is to learn

$$P \approx P_{\mathrm{GT}}$$

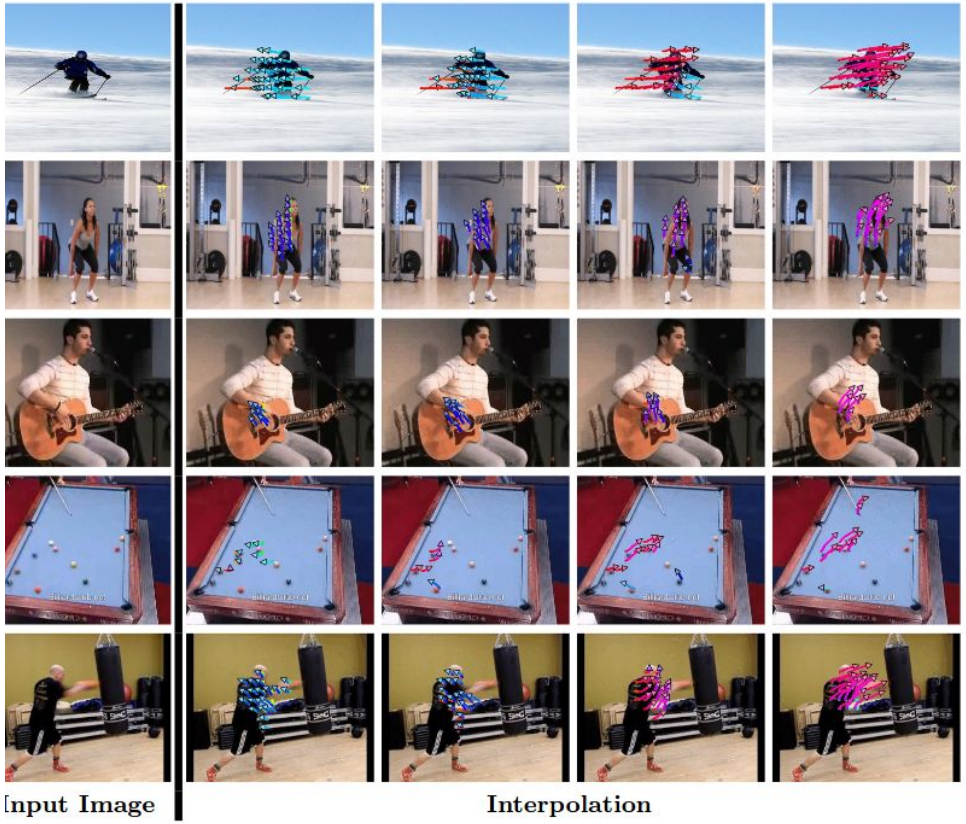
Collection of Samples $\mathcal{X}$



Variational Autoencoders - Applications

- Image Compression
- Predicting future from static images
- Graphic designing
- etc...

Context-adaptive Entropy Model for End-to-end Optimized Image Compression, Lee et al. ICLR 2019.



An Uncertain Future: Forecasting from Static Images using Variational Autoencoders, Walker et al. ECCV 2016

Latent Variable Models

- Say, that we have a vector $\mathbf{z}$ in a high-dimensional space $\mathcal{Z}$
- We can easily sample $\mathbf{z}$ according to a probability distribution function (PDF) $\mathbf{P}(\mathbf{z})$
- Say that we have a family of deterministic functions $\mathbf{f}(\mathbf{z};\Theta)$
- Θ are the parameters of the function $\mathbf{f}$ in some space $\boldsymbol{\Theta}$

$$\mathbf{f}: \mathcal{Z} \times \boldsymbol{\Theta} \square \mathcal{X}$$

Objective: optimize Θ such that sampling $\mathbf{z}$ via $\mathbf{P}(\mathbf{z})$ will result in real looking images.

Latent Variable Models

- Say, that we have a vector **z** in a high-dimensional space $\mathcal{Z}$
- We can easily sample **z** according to a probability distribution function (PDF) $\mathbf{P(z)}$
- Say that we have a family of deterministic functions $\mathbf{f(z;\Theta)}$
- $\mathbf{\Theta}$ are the parameters of the function $\mathbf{f}$ in some space $\boldsymbol{\Theta}$

$$\mathbf{f}: \mathcal{Z} \times \boldsymbol{\Theta} \square \mathcal{X}$$

Objective: optimize $\mathbf{\Theta}$ such that sampling **z** via $\mathbf{P(z)}$ will result in real looking images.

$$\mathbf{P}(X) = \int \mathbf{P}(X|\mathbf{z};\mathbf{\Theta})\mathbf{P}(\mathbf{z})\, \partial\mathbf{z}$$

$$\mathbf{P}(X|\mathbf{z};\mathbf{\Theta}) = \mathbf{N}(X \mid \mathbf{f}(\mathbf{z};\mathbf{\Theta}), \boldsymbol{\sigma}(\mathbf{z};\mathbf{\Theta})^2 * \mathbf{I})$$

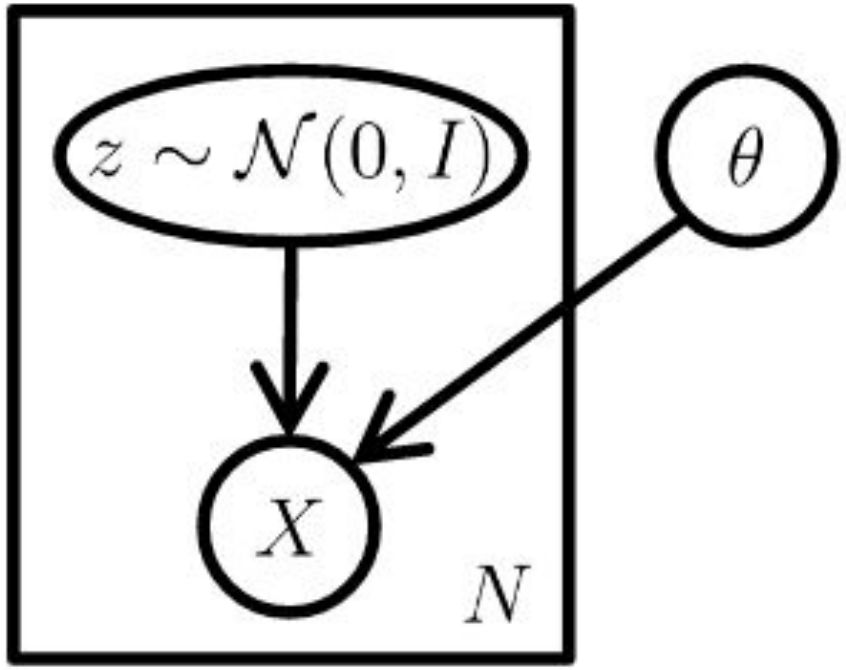
Variational Autoencoders - Challenges

Objective: $\mathbf{P}(\mathrm{X}) = \int \mathbf{P}(\mathrm{X}|\mathbf{z};\boldsymbol{\Theta})\mathbf{P}(\mathbf{z})\, \partial\mathbf{z}$

Two main problems:

a. How to define the latent variables **z** (that is decide what information they represent).
b. How to deal with the integral over **z**.

Latent Variable Models - Graphical Model



a. How to define the latent variables **z**

Objective: $\mathbf{P}(\mathrm{X}) = \int \mathbf{P}(\mathrm{X}|\mathbf{z};\boldsymbol{\Theta})\mathbf{P}(\mathbf{z})\, \partial\mathbf{z}$

VAEs solve this by assuming that **z** is drawn from a normal distribution with zero mean and identity covariance:

$$\mathbf{P}(\mathbf{z}) = \mathbf{N}(\mathbf{z} \mid 0, \mathbf{I})$$

How is this possible? How can a simple distribution $\mathbf{P}(\mathbf{z}) = \mathbf{N}(\mathbf{z} \mid 0, \mathbf{I})$ can lead to the generation of complex data samples?

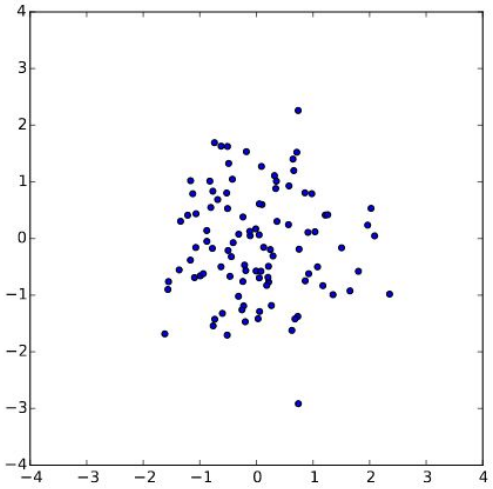
a. How to define the latent variables **z**

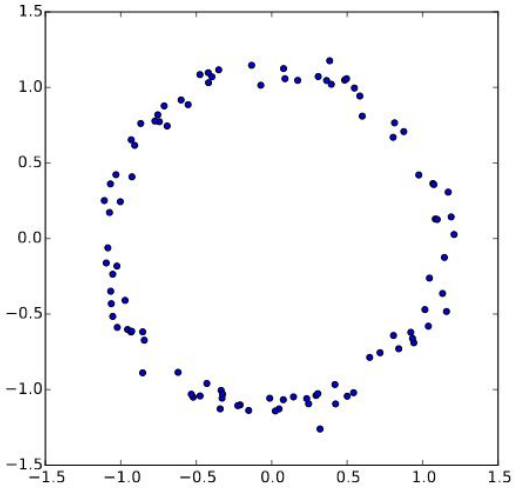
How can a simple distribution $\mathbf{P(z) = N(z \mid 0, I)}$ can lead to the generation of complex data samples?

Remember $\mathbf{P(X|z;\Theta) = N(X \mid f(z;\Theta), \sigma(z;\Theta)^2 * I)}$?

If $\mathbf{f(z;\Theta)}$ is a powerful function approximator, generating data samples is possible!

NEURAL NETWORK DECODER





$\mathbf{f(z) = z/10 + z/||z||}$

a. How to define the latent variables **z**

Objective: $\mathbf{P}(\mathrm{X}) = \int \mathbf{P}(\mathrm{X}|\mathbf{z};\boldsymbol{\Theta})\mathbf{P}(\mathbf{z})\, \partial\mathbf{z}$

Not all **z** will lead to the generation of meaningful samples.
During training VAEs sample **z** efficiently by using another function $Q(\mathbf{z} \mid \mathrm{X})$

Lets us compute $E_{\mathrm{z}\sim\mathrm{Q}}\mathbf{P}(\mathrm{X}|\mathbf{z};\boldsymbol{\Theta})$ relatively easily

NEURAL NETWORK ENCODER

Relating $E_{z\sim Q}\mathbf{P}(X|\mathbf{z};\Theta)$ and $\mathbf{P}(X)$

The two distributions are compared using Kullback-Leibler divergence (KL divergence)

$$\mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z}|X)] = E_{z\sim Q}[\log\boldsymbol{Q}(\mathbf{z} \mid X) - \log\mathbf{P}(\mathbf{z}|X)]$$

From Bayes rule: $\log\mathbf{P}(\mathbf{z}|X) = \log\mathbf{P}(X|\mathbf{z}) + \log\mathbf{P}(\mathbf{z}) - \log\mathbf{P}(X)$

Substituting

$$\mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z}|X)] = E_{z\sim Q}[\log\boldsymbol{Q}(\mathbf{z} \mid X) - \log\mathbf{P}(X|\mathbf{z}) - \log\mathbf{P}(\mathbf{z})] + \log\mathbf{P}(X)$$

Rearranging

$$\log\mathbf{P}(X) - \mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z}|X)] = E_{z\sim Q}[\log\mathbf{P}(X|\mathbf{z})] - \mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z})]$$

Does not depend on **z**

Understanding the Objective

Can be optimized using SGD!

$$\log \mathbf{P}(X) - KL[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z}|X)] = E_{z\sim Q}[\log \mathbf{P}(X|\mathbf{z})] - KL[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z})]$$

What we want to maximize

Ensures that the encoder outputs **z** that can compute X

Objective for the **decoder**

Objective for the **encoder**

Understanding the Objective

Assumption: High capacity

$$\log \mathbf{P}(X) - KL[\boldsymbol{Q}(\mathbf{z} \mid X) \,||\, \mathbf{P}(\mathbf{z}|X)] = E_{z\sim Q}[\log \mathbf{P}(X|\mathbf{z})] - KL[\boldsymbol{Q}(\mathbf{z} \mid X) \,||\, \mathbf{P}(\mathbf{z})]$$

Can't be computed analytically (not tractable)

Understanding the Objective

$$\log \mathbf{P}(X) - \mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z}|X)] = E_{z\sim Q}[\log \mathbf{P}(X|\mathbf{z})] - \mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,\|\, \mathbf{P}(\mathbf{z})]$$

Nearly zero! Thus we can use $\boldsymbol{Q}(\mathbf{z} \mid X)$ to compute $\mathbf{P}(\mathbf{z}|X)$

Optimizing the Objective

$$\text{R.H.S} = E_{z\sim Q}[\log \mathbf{P}(X|\mathbf{z})] - \mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,||\, \mathbf{P}(\mathbf{z})]$$

$$\boldsymbol{Q}(\mathbf{z} \mid X) = \mathbf{N}(\mathbf{z}|\boldsymbol{\mu}(X;\boldsymbol{\varphi}), \boldsymbol{\Sigma}(X;\boldsymbol{\varphi}))$$

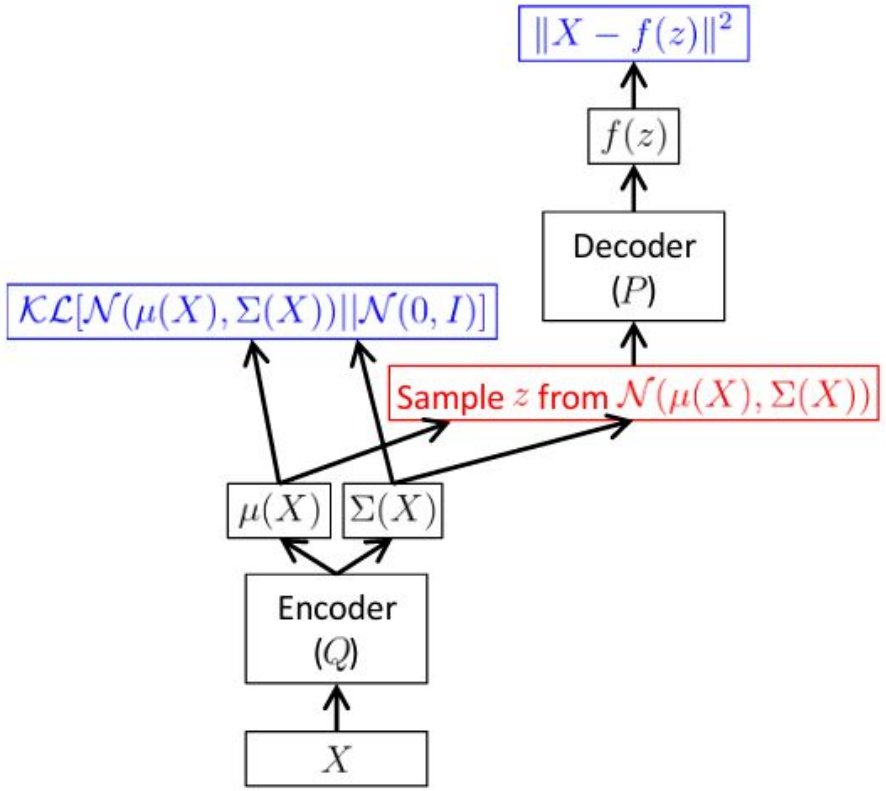
Parameterized deterministic functions

$\boldsymbol{\Sigma}(X;\boldsymbol{\varphi})$ is constrained to be a diagonal matrix to compute the KL divergence efficiently

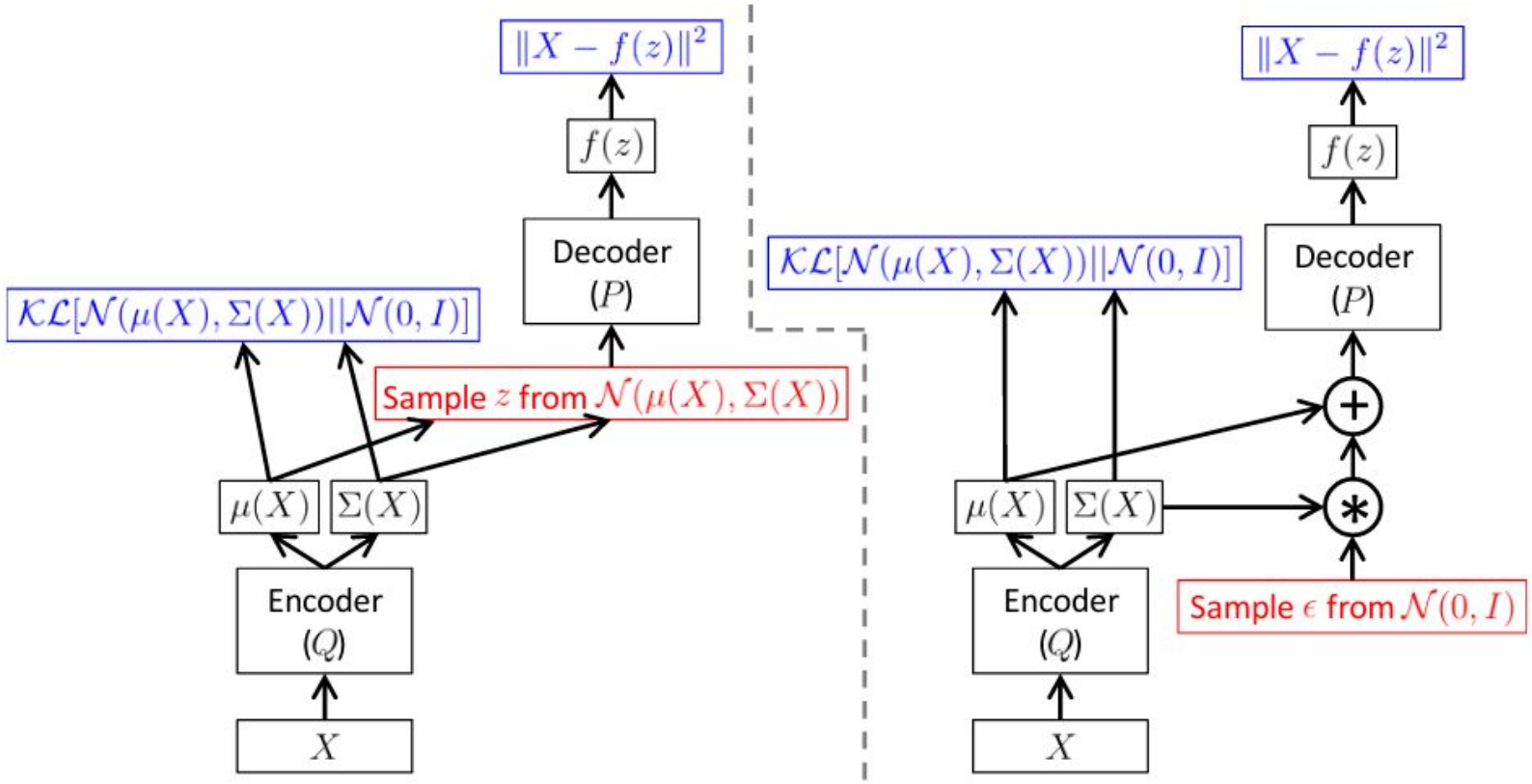
$$\mathrm{KL}[\boldsymbol{Q}(\mathbf{z} \mid X) \,||\, \mathbf{P}(\mathbf{z})] = \mathrm{KL}[\mathbf{N}(\mathbf{z}|\boldsymbol{\mu}(X;\boldsymbol{\varphi}), \boldsymbol{\Sigma}(X;\boldsymbol{\varphi})) \,||\, \mathbf{N}(\mathbf{z} \mid 0, \mathbf{I})]$$
$$= 0.5(\mathrm{tr}(\boldsymbol{\Sigma}(X;\boldsymbol{\varphi})) + \boldsymbol{\mu}(X;\boldsymbol{\varphi})^{T}\boldsymbol{\mu}(X;\boldsymbol{\varphi}) - k - \log(\det(\boldsymbol{\Sigma}(X;\boldsymbol{\varphi}))))$$

Number of dimensions

Back-propagation - Reparameterization Trick



Back-propagation - Reparameterization Trick



Moving to representation learning….

Note: The notations for equations and the figures of the AVT (ICCV’19) paper have been adjusted to align with the VAE tutorial.

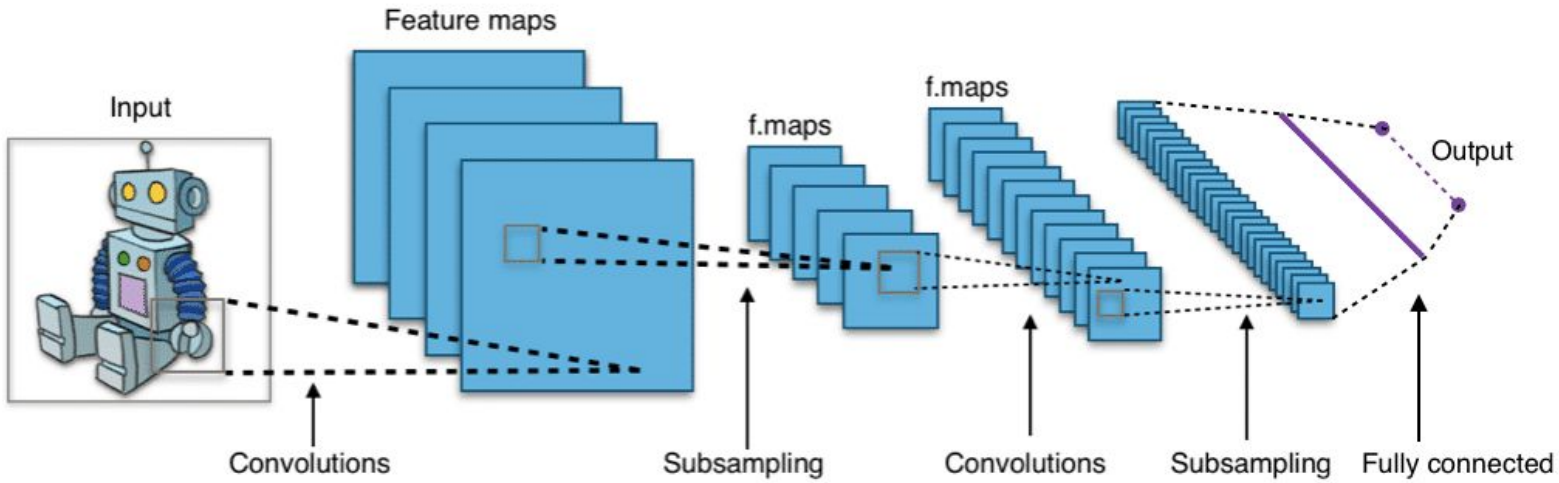
Transformation Equivariant Representations: Convolutional Neural Networks

Training data: $(x,y) \sim X$

Model output: $\hat{y} = f(aug(x))$

Objective fnc: $min[\Sigma loss(\hat{y},y)]$

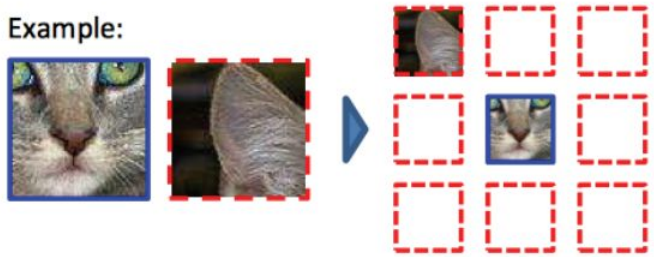
data augmentation



Can TERs be a criteria for learning unsupervised representations?

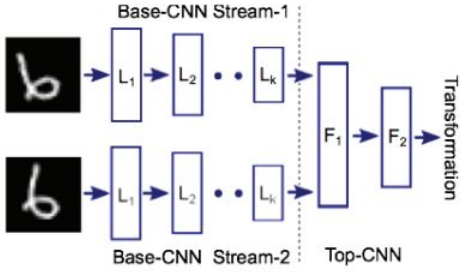
Unsupervised Representation Learning

Single Images



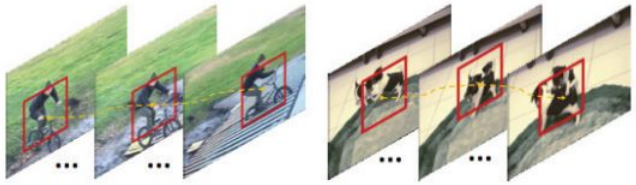
Doersch et al. 2015

Motion



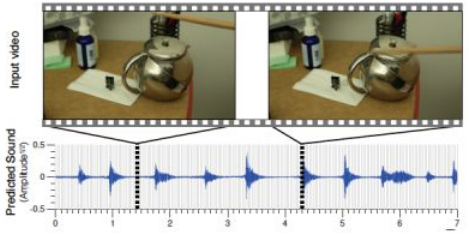
Agrawal et al. 2015

Videos

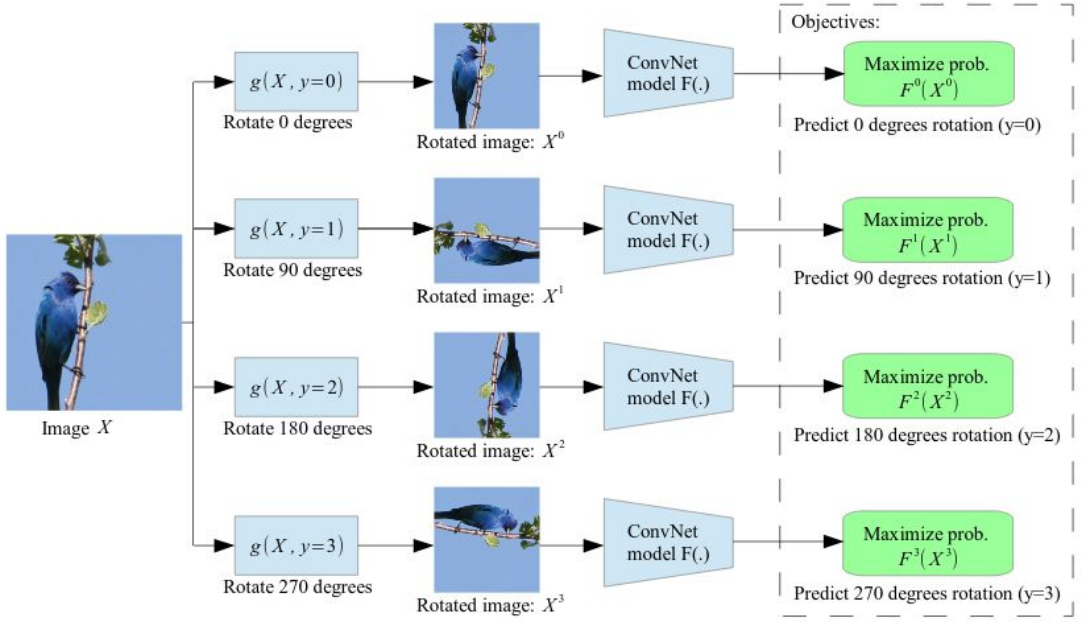


Wang et al. 2015

Sound



Owens et al. 2016



Gidaris et al. 2018

Use of non-visual signals, intrinsically correlated to the image, as a form to supervise visual feature learning.

Transformation Equivariant Representations

Parameterized Transformation: $\mathbf{t} \sim \mathrm{p}(\mathbf{t})$
Image: $\mathbf{x} \sim \mathrm{p}(\mathbf{x})$
Transformed Image: $\mathbf{t}(\mathbf{x})$

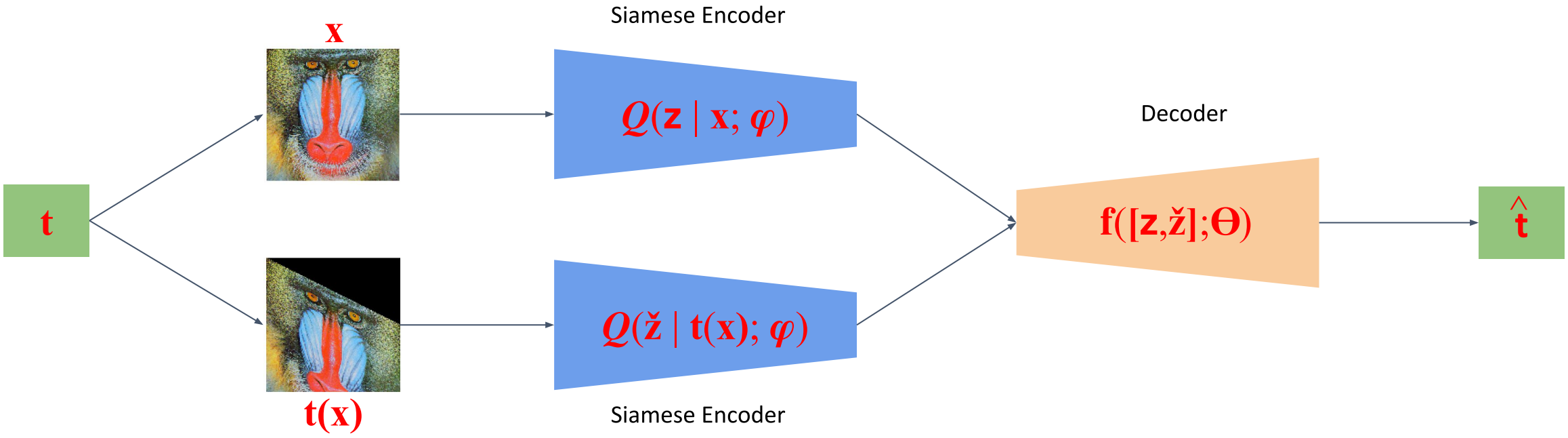
Model with parameters $\boldsymbol{\varphi}$

$Q(\mathbf{z} \mid \mathbf{t}(\mathbf{x}); \boldsymbol{\varphi})$ → Representation of the transformed image

$Q(\check{\mathbf{z}} \mid \mathbf{x}; \boldsymbol{\varphi})$ → Representation of the original image

Objective: Use transformation equivariant property as a supervisory signal for learning unsupervised representations $Q(\mathbf{x}; \boldsymbol{\varphi})$.

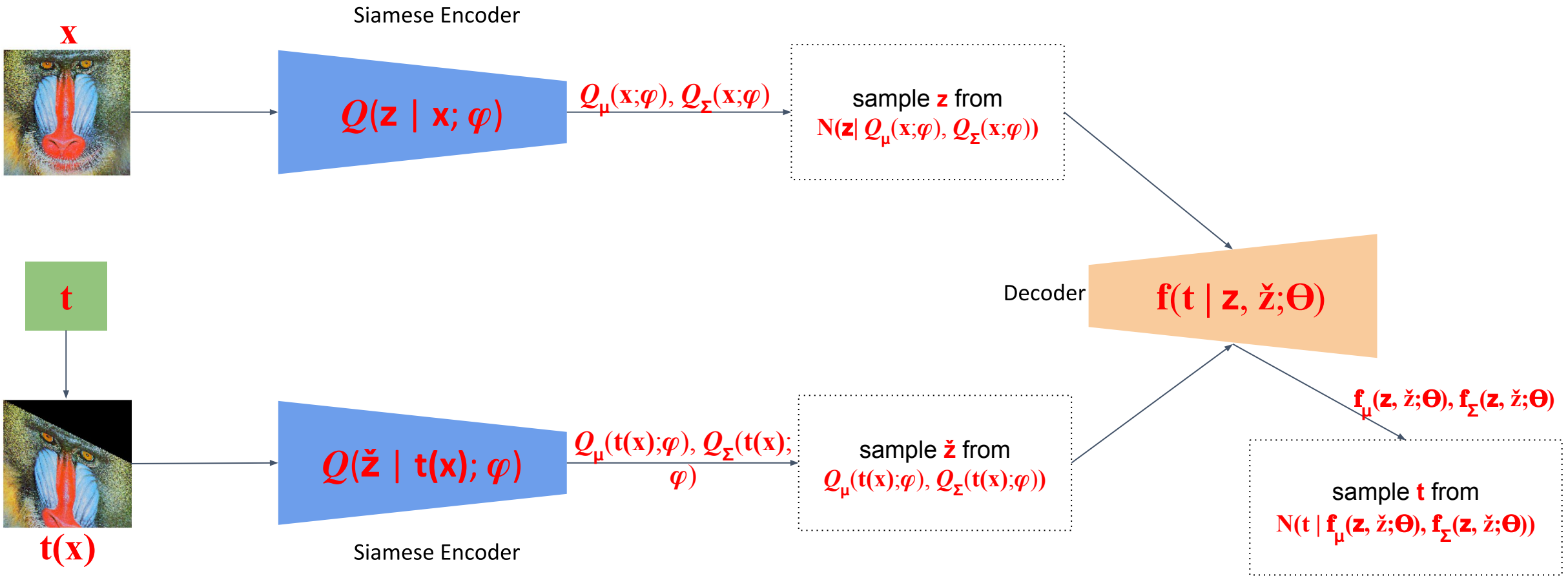
AET: Auto-Encoders for Predicting Transformations



Objective fnc: $\min[E_{\mathbf{t} \sim p(\mathbf{t}), \mathbf{x} \sim p(\mathbf{x})} \mathrm{loss}(\hat{\mathbf{t}}, \mathbf{t})]$

L2 between projective transformation metrics

AVT: Auto-encoding Variational Transformations



AVT: Optimization

$$I(\mathbf{t},\check{\mathbf{z}}|\mathbf{z}) = \mathbf{H}(\mathbf{t}|\mathbf{z}) - \mathbf{H}(\mathbf{t}|\check{\mathbf{z}}, \mathbf{z})$$

mutual information

conditional entropies (from the definition of mutual information)

$$I(\mathbf{t};\check{\mathbf{z}}|\mathbf{z}) = \mathbf{H}(\mathbf{t}|\mathbf{z}) + E_{\mathbf{z},\check{\mathbf{z}}\sim Q(;\boldsymbol{\varphi}),\ \mathbf{t}\sim p(\mathbf{t})}[\log Q(\mathbf{t}|\check{\mathbf{z}}, \mathbf{z}; \boldsymbol{\varphi})]$$

from the definition of conditional entropy

$$I(\mathbf{t};\check{\mathbf{z}}|\mathbf{z}) = \mathbf{H}(\mathbf{t}|\mathbf{z}) + E_{\mathbf{z},\check{\mathbf{z}}\sim Q(;\boldsymbol{\varphi}),\ \mathbf{t}\sim p(\mathbf{t})}[\log \mathbf{f}(\mathbf{t} \mid \mathbf{z}, \check{\mathbf{z}};\boldsymbol{\Theta})] + E_{\mathbf{z},\check{\mathbf{z}}\sim Q(;\boldsymbol{\varphi}),\ \mathbf{t}\sim p(\mathbf{t})}\mathrm{KL}[Q(\mathbf{t}|\check{\mathbf{z}}, \mathbf{z}; \boldsymbol{\varphi}) \,||\, \mathbf{f}(\mathbf{t} \mid \mathbf{z}, \check{\mathbf{z}};\boldsymbol{\Theta})]$$

Always non-negative

$$I(\mathbf{t};\check{\mathbf{z}}|\mathbf{z}) \geq \mathbf{H}(\mathbf{t}|\mathbf{z}) + E_{\mathbf{z},\check{\mathbf{z}}\sim Q(;\boldsymbol{\varphi}),\ \mathbf{t}\sim p(\mathbf{t})}[\log \mathbf{f}(\mathbf{t} \mid \mathbf{z}, \check{\mathbf{z}};\boldsymbol{\Theta})]$$

AVT: Optimization

$$I(\mathbf{t};\check{\mathbf{z}}|\mathbf{z}) \geq \mathbf{H}(\mathbf{t}|\mathbf{z}) + E_{\mathbf{z},\check{\mathbf{z}}\sim Q(;\boldsymbol{\varphi}),\ \mathbf{t}\sim p(\mathbf{t})}[\log \mathbf{f}(\mathbf{t} \mid \mathbf{z}, \check{\mathbf{z}};\boldsymbol{\Theta})]$$

Does not depend on
model parameters

Objective: $\max_{\boldsymbol{\varphi},\boldsymbol{\Theta}} E_{\mathbf{z},\check{\mathbf{z}}\sim Q(;\boldsymbol{\varphi}),\ \mathbf{t}\sim p(\mathbf{t})}[\log \mathbf{f}(\mathbf{t} \mid \mathbf{z}, \check{\mathbf{z}};\boldsymbol{\Theta})]$

$$\max_{\theta,\varphi} \frac{1}{n} \sum_{i=1}^{n} \mathcal{N}(t^i | f_\mu(z, \hat{z}; \theta), f_\sigma(z, \hat{z}; \theta))$$

AVT: Results on CIFAR-10 dataset

Table 2: Error rates of different classifiers trained on top of the learned representations on CIFAR 10, where n-FC denotes a classifier with n fully connected layers and conv denotes the third NIN block as a convolutional classifier. Two AET variants are chosen for a fair direct comparison since they are based on the same architecture as the AVT and have outperformed the other unsupervised representations before [37].

	1 FC	2 FC	3 FC	conv
AET-affine [37]	17.16	9.77	10.16	8.05
AET-project [37]	16.65	9.41	9.92	7.82
(Ours) AVT	**16.19**	**8.96**	**9.55**	**7.75**

Deterministic models

Probabilistic models used projective transformations for training

AVT: Results on ImageNet dataset

TABLE 6: Top-1 accuracy with linear layers on ImageNet. AlexNet is used as backbone to train the unsupervised models under comparison. A $1,000$-way linear classifier is trained upon various convolutional layers of feature maps that are spatially resized to have about $9,000$ elements. Fully supervised and random models are also reported to show the upper and the lower bounds of unsupervised model performances. Only a single crop is used and no dropout or local response normalization is used during testing, except the models denoted with * where ten crops are applied to compare results.

Method	Conv1	Conv2	Conv3	Conv4	Conv5
ImageNet labels(Upper Bound)	19.3	36.3	44.2	48.3	50.5
Random (Lower Bound)	11.6	17.1	16.9	16.3	14.1
Random rescaled [38]	17.5	23.0	24.5	23.2	20.6
Context [21]	16.2	23.3	30.2	31.7	29.6
Context Encoders [39]	14.1	20.7	21.0	19.8	15.5
Colorization [36]	12.5	24.5	30.4	31.5	30.3
Jigsaw Puzzles [20]	18.2	28.8	34.0	33.9	27.1
BIGAN [15]	17.7	24.5	31.0	29.9	28.0
Split-Brain [40]	17.7	29.3	35.4	35.2	32.8
Counting [40]	18.0	30.6	34.3	32.5	25.7
RotNet [23]	18.8	31.7	38.7	38.2	36.5
AET-project	19.2	32.8	40.6	39.7	37.7
AVT-project	**19.5**	**33.6**	**41.3**	**40.3**	**39.1**
DeepCluster* [37]	13.4	32.3	41.0	39.6	38.2
AET-project*	19.3	35.4	44.0	43.6	42.4
AVT-project*	**20.9**	**36.1**	**44.4**	**44.3**	**43.5**

AVT: Results on Places dataset

TABLE 7: Top-1 accuracy on the Places dataset. A 205-way logistic regression classifier is trained on top of various layers of feature maps that are spatially resized to have about 9,000 elements. All unsupervised features are pre-trained on the ImageNet dataset, and then frozen when training the logistic regression classifiers with Places labels. We also compare with fully-supervised networks trained with Places Labels and ImageNet labels, as well as with random models. The highest accuracy values are in bold and the second highest accuracy values are underlined.

Method	Conv1	Conv2	Conv3	Conv4	Conv5
Places labels(Upper Bound) [41]	22.1	35.1	40.2	43.3	44.6
ImageNet labels	22.7	34.8	38.4	39.4	38.7
Random (Lower Bound)	15.7	20.3	19.8	19.1	17.5
Random rescaled [38]	21.4	26.2	27.1	26.1	24.0
Context [21]	19.7	26.7	31.9	32.7	30.9
Context Encoders [39]	18.2	23.2	23.4	21.9	18.4
Colorization [36]	16.0	25.7	29.6	30.3	29.7
Jigsaw Puzzles [20]	23.0	31.9	35.0	34.2	29.3
BIGAN [15]	22.0	28.7	31.8	31.3	29.7
Split-Brain [40]	21.3	30.7	34.0	34.1	32.5
Counting [40]	**23.3**	**33.9**	36.3	34.7	29.6
RotNet [23]	21.5	31.0	35.1	34.6	33.7
AET-project	22.1	32.9	37.1	36.2	34.7
AVT-project	22.3	33.1	**37.8**	**36.7**	**35.6**

Thank you!

@yash0307

patelyas@fel.cvut.cz

Homepage: https://yash0307.github.io