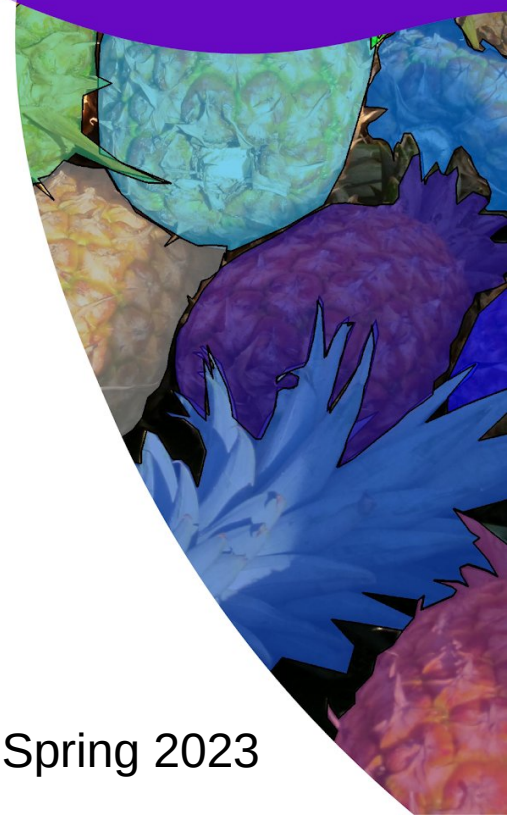
NEW YORK UNIVERSITY

SSL and Joint Embedding Architectures

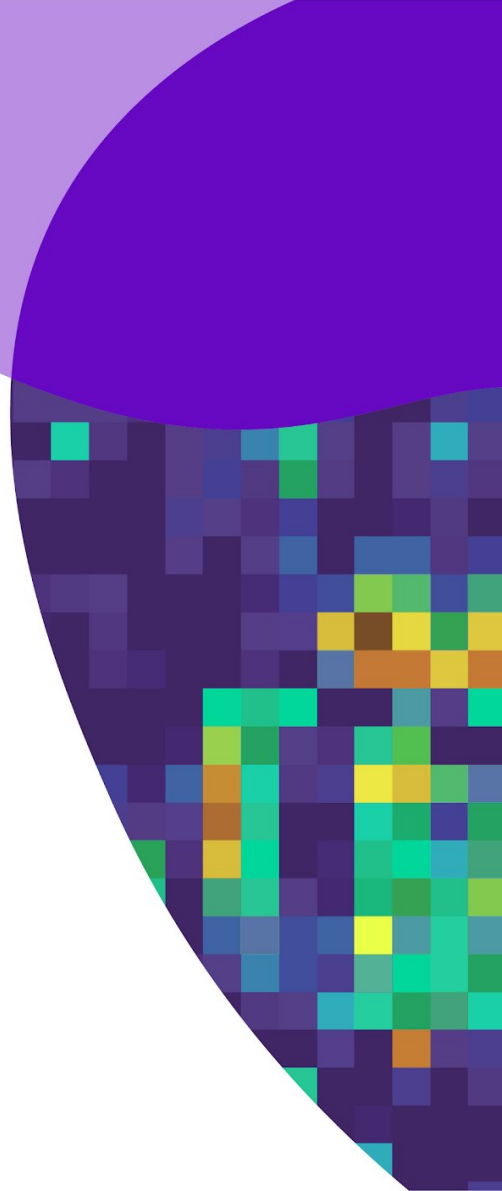
Yann LeCun
NYU - Courant Institute & Center for Data Science
Facebook AI Research
http://yann.lecun.com



Deep Learning, NYU, Spring 2023

FACEBOOK AI

Self-Supervised Learning

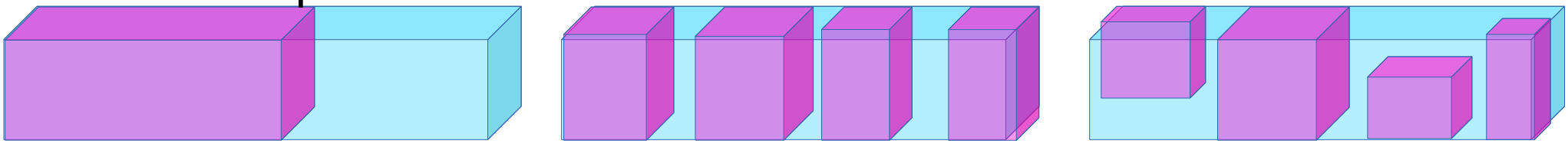


Capture dependencies between inputs.

Self-Supervised Learning = Learning to Fill in the Blanks

- **Reconstruct the input or Predict missing parts of the input.**

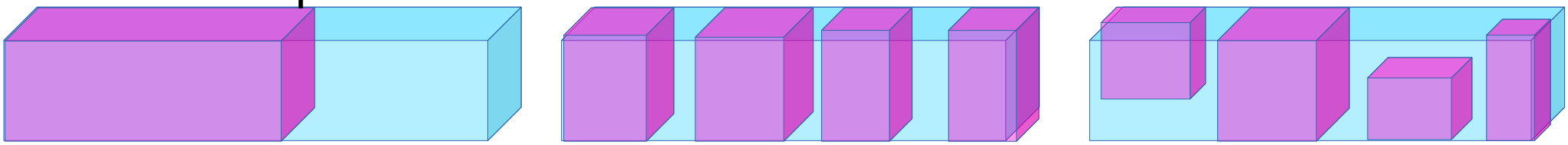
time or space →



Self-Supervised Learning = Learning to Fill in the Blanks

- **Reconstruct the input or Predict missing parts of the input.**

time or space →

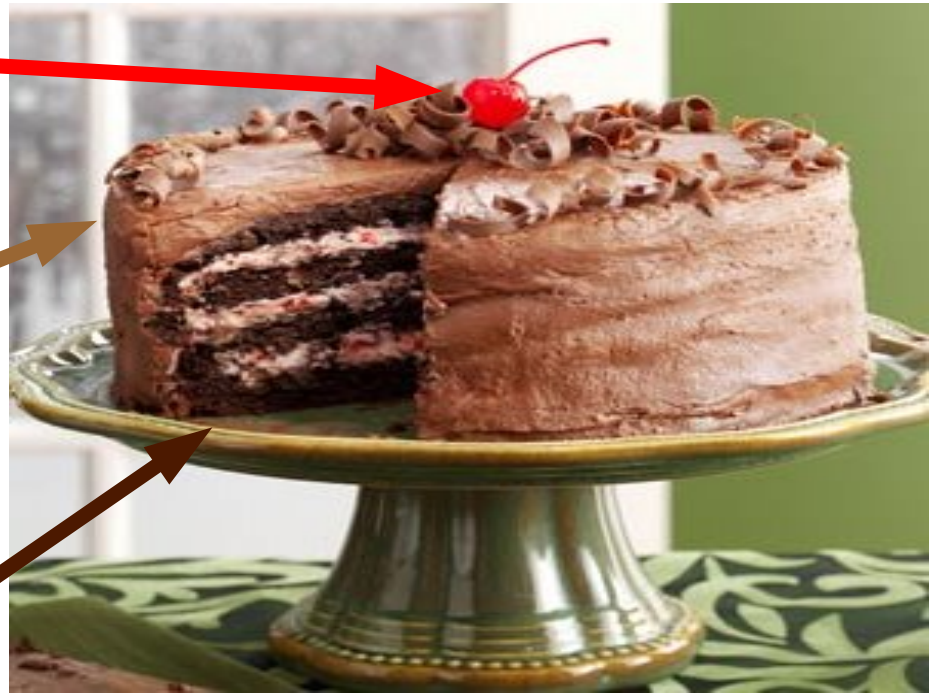


Two Uses for Self-Supervised Learning

- **1. Learning hierarchical representations of the world**
  - SSL pre-training precedes a supervised or RL phase
- **2. Learning predictive (forward) models of the world**
  - Learning models for Model-Predictive Control, policy learning for control, or model-based RL.
- **Question: how to represent uncertainty & multi-modality in the prediction?**

Learning Paradigms: information content per sample

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

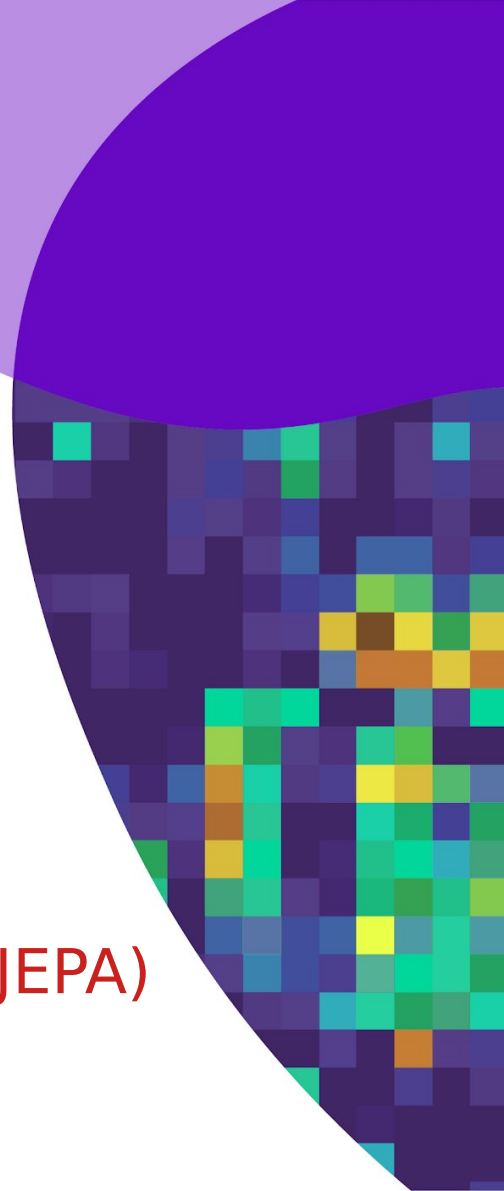


NEW YORK UNIVERSITY

EBM Architectures

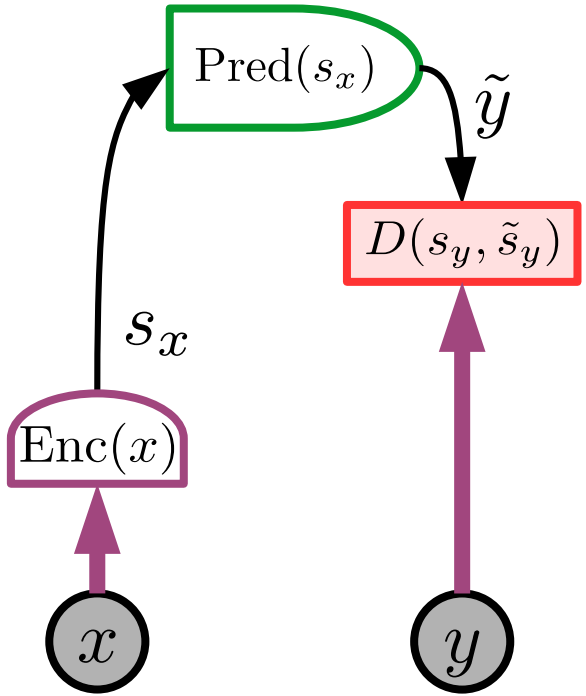
for multimodal prediction

Latent variable models
Joint embedding architectures
Joint Embedding Predictive Architecture (JEPA)

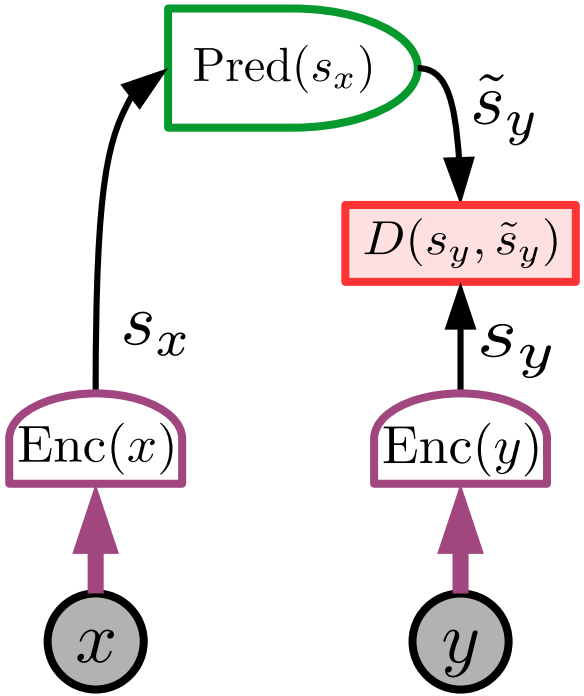


EBM Architectures: Generative vs Joint Embedding

- **Generative: predicts y**
- **Joint Embedding: predicts an abstract representation of y**



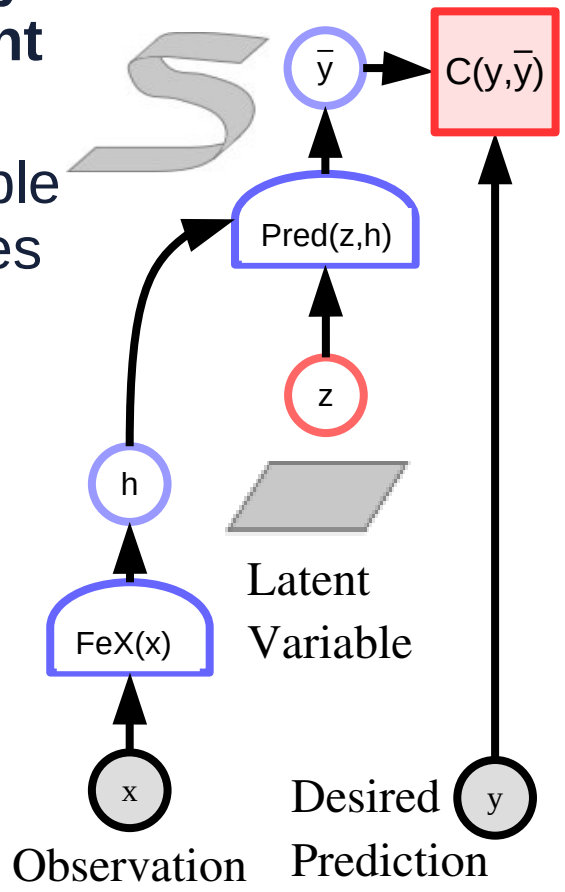
a) Generative Architecture

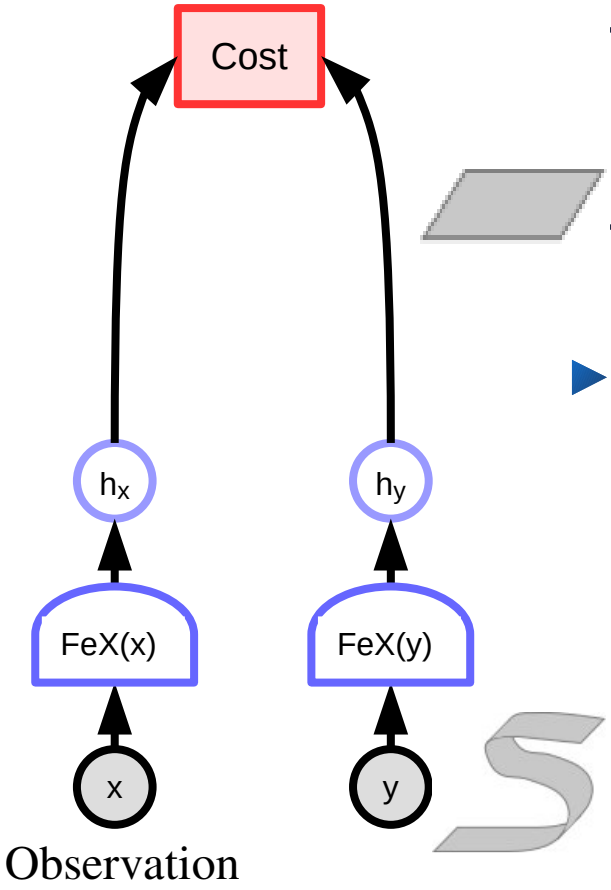


b) Joint Embedding Architecture

Latent-Variable Generative EBM / Joint Embedding EBM

- **Multimodality through latent variable**
  - Latent Variable parameterizes the set of plausible predictions

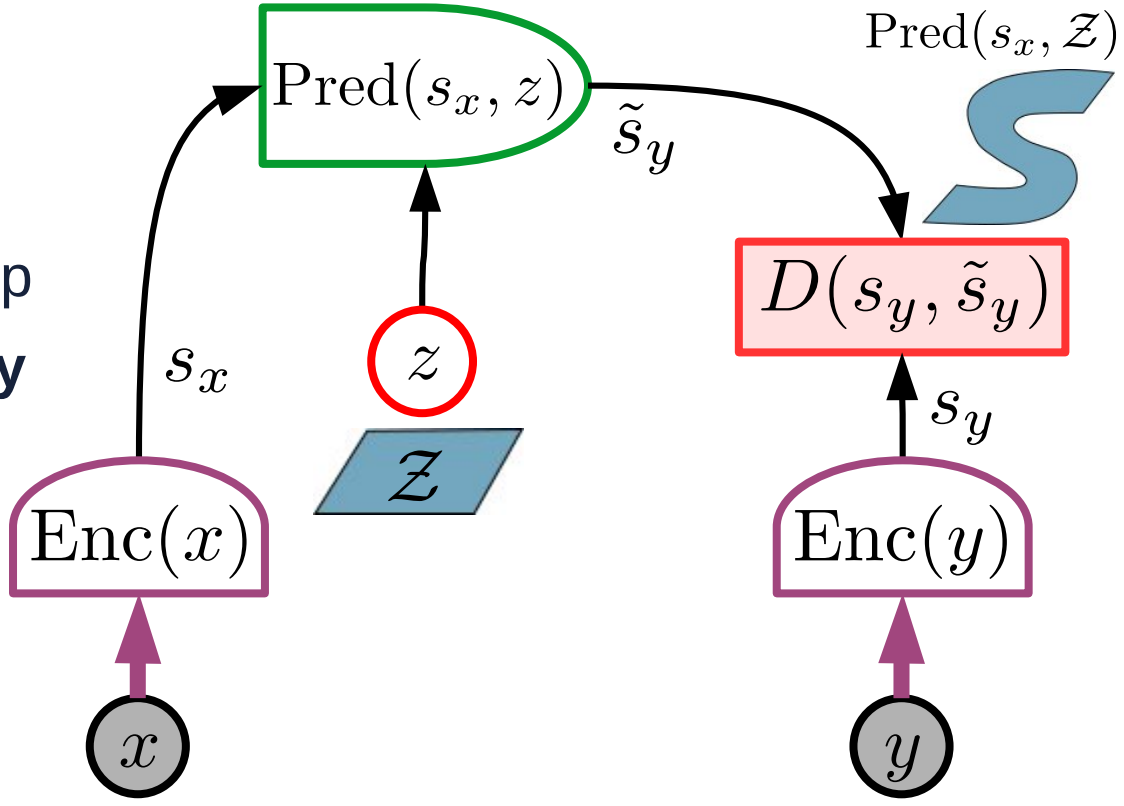




- **Multimodality through invariance properties of the right branch.**
  - Multiple y will produce the same representation and will have the same energy.

Joint Embedding Predictive Architecture (JEPA)

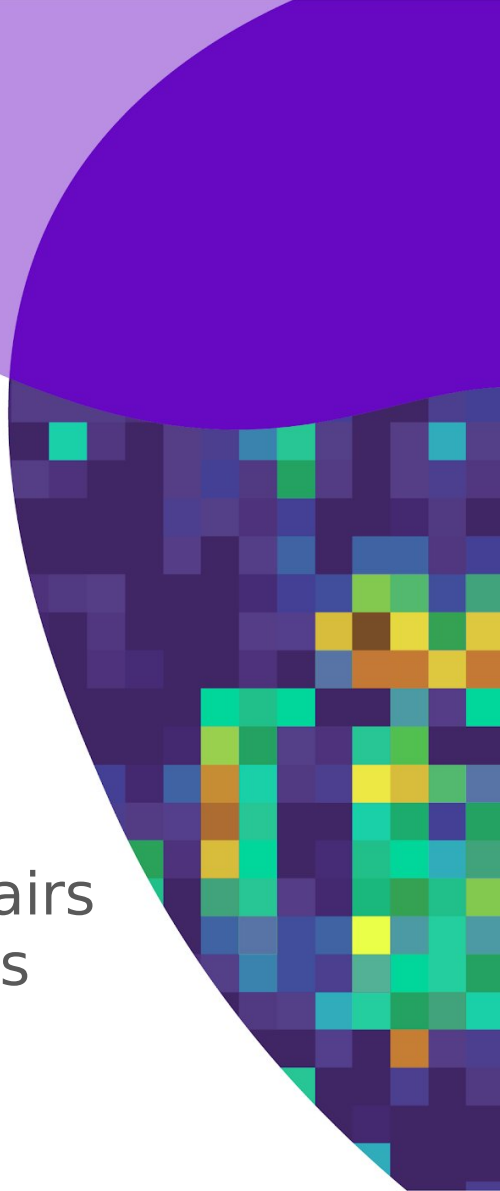
- **Computes abstract representations for x and y**
- **Makes predictions in representation space**
  - Can use a latent variable to help
- **Does not need to predict every details of y**
  - Enc(y) can eliminate irrelevant details through invariances
- **Tries to make the representations predictable from each other.**



FACEBOOK AI

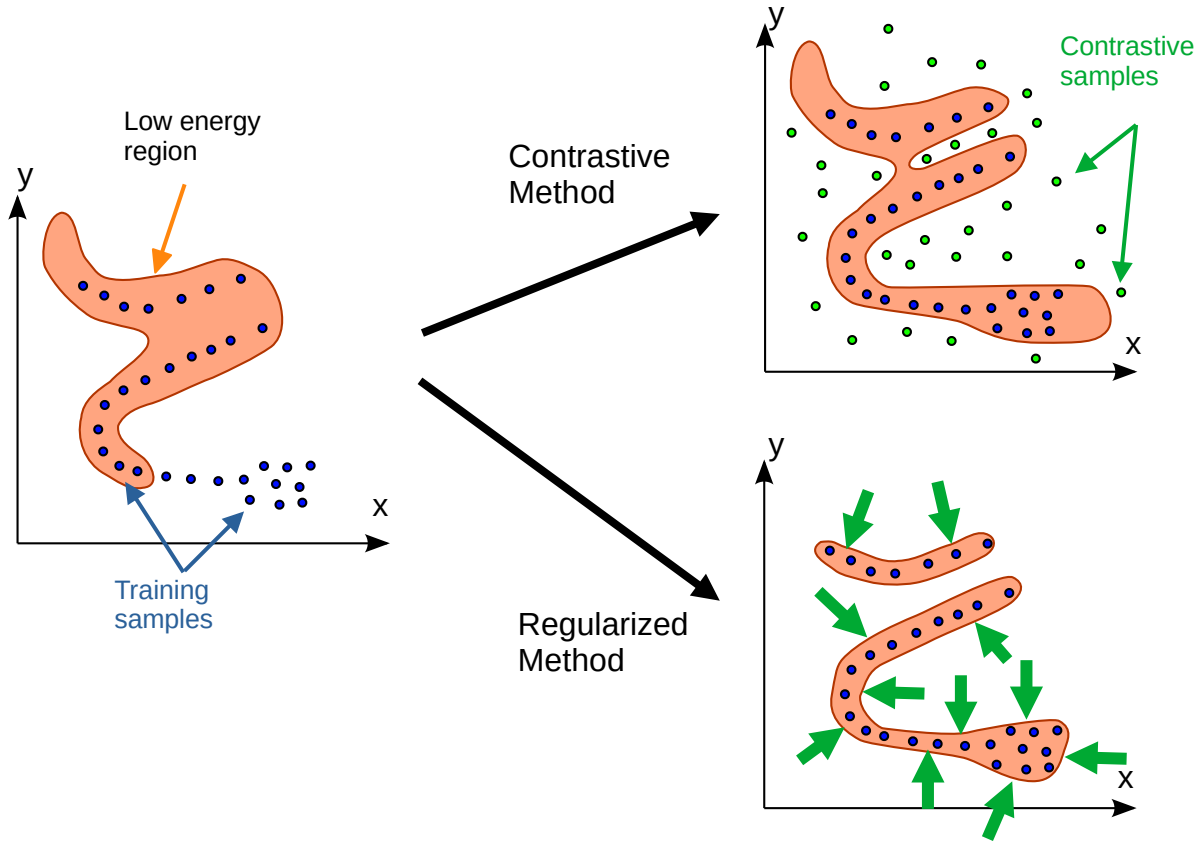
Contrastive Methods for joint embedding architectures

Push down on the energy of compatible sample pairs
Pull up on the energy of incompatible sample pairs



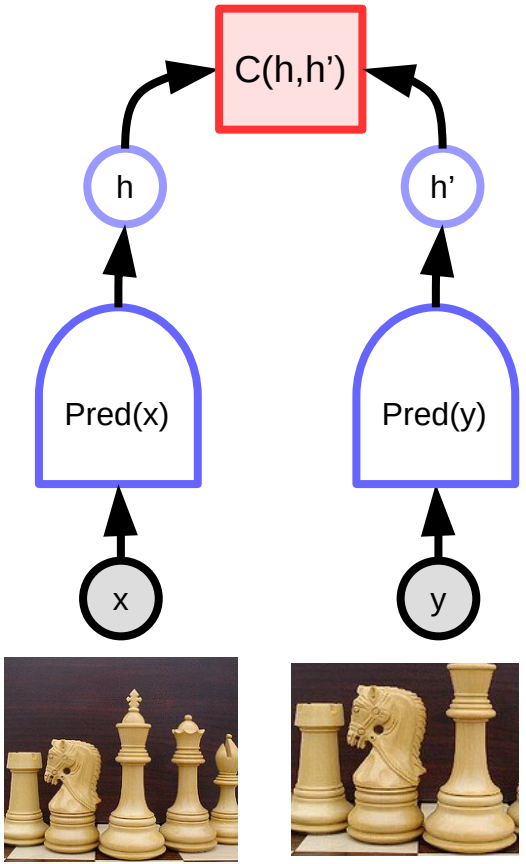
EBM Training: Contrastive vs Regularized methods

- **Contrastive methods**
  - Push down on energy of training samples
  - Pull up on energy of suitably-generated contrastive samples
  - Scales very badly with dimension
- **Regularized Methods**
  - Regularizer minimizes the volume of space that can take low energy

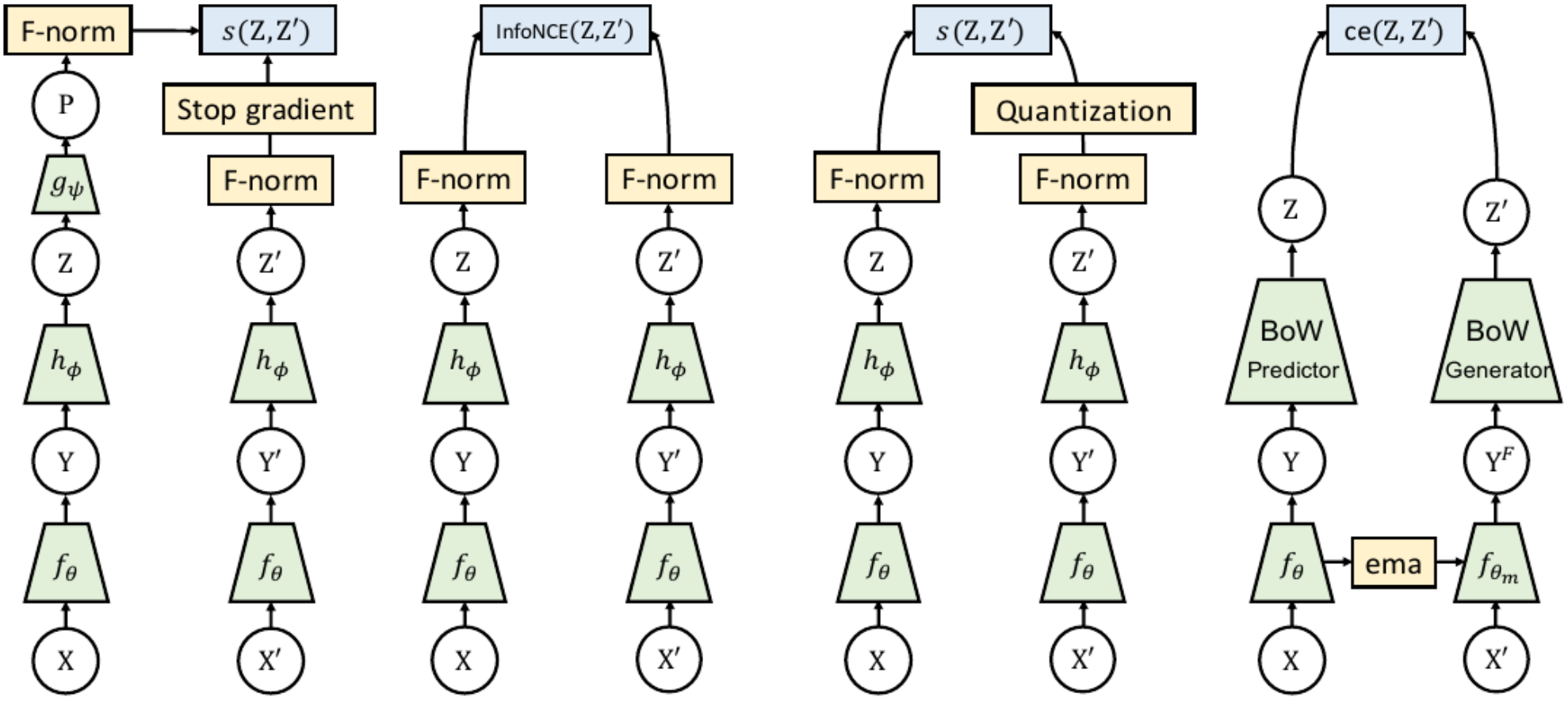


Joint Embedding Architectures

- **Distance measured in feature space**
- **Multiple “predictions” through feature invariance**
- **Siamese nets, metric learning**
  - [Bromley NIPS’93] [Chopra CVPR’05] [Hadsell CVPR’06]
- **Advantage: no pixel-level reconstruction**
- **Difficulty: <in a few slides>**
- **Many successful examples for image recognition:**
  - DeepFace [Taigman et al. CVPR 2014]
  - PIRL [Misra et al. Arxiv:1912.01991]
  - MoCo [He et al. Arxiv:1911.05722]
  - SimCLR [Chen et al. Arxiv:2002.05709]
  - …..



Contrastive Joint Embedding Methods



(e) SimSiam (f) SimCLR (g) SwAV (h) OBoW

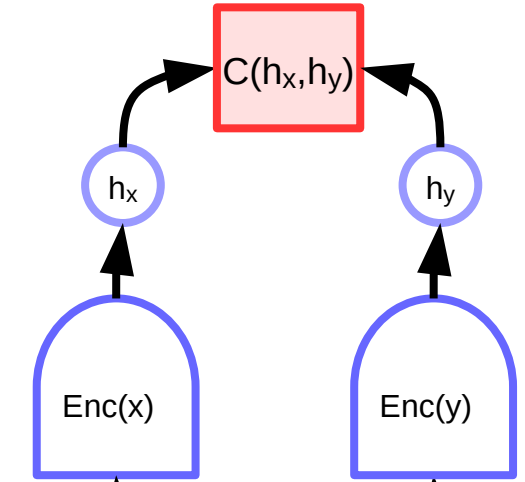
Contrastive Joint Embedding

$$F(x, y) = C(\text{Enc}(x), \text{Enc}(y))$$

- **Siamese nets, metric learning**
- **Two identical networks with shared weights**
  - Signature verification: [Bromley NIPS'93],
  - Face verification [Chopra CVPR'05]
  - Face reco, DeepFace [Taigman et al. CVPR 2014]
  - Video feature learning [Taylor CVPR 2011]
  - Use square-square or square-exp loss

$$\mathcal{L}(x, y, \hat{y}, w) = \left([F_w(x, y)]^+\right)^2 + \left([m - F_w(x, \hat{y})]^+\right)^2$$

- **Advantages:**
  - no pixel-level reconstruction
  - Learns a similarity metric
  - Multimodality through encoder invariance

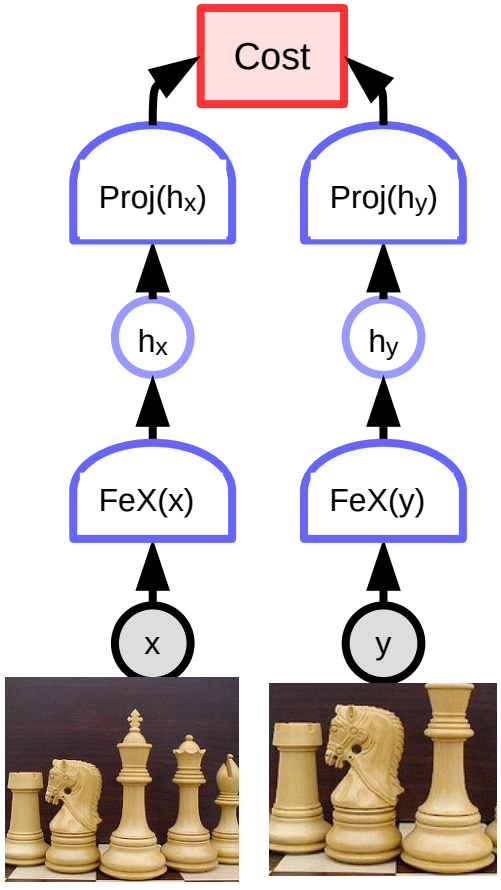


Positive pair:
Make F small

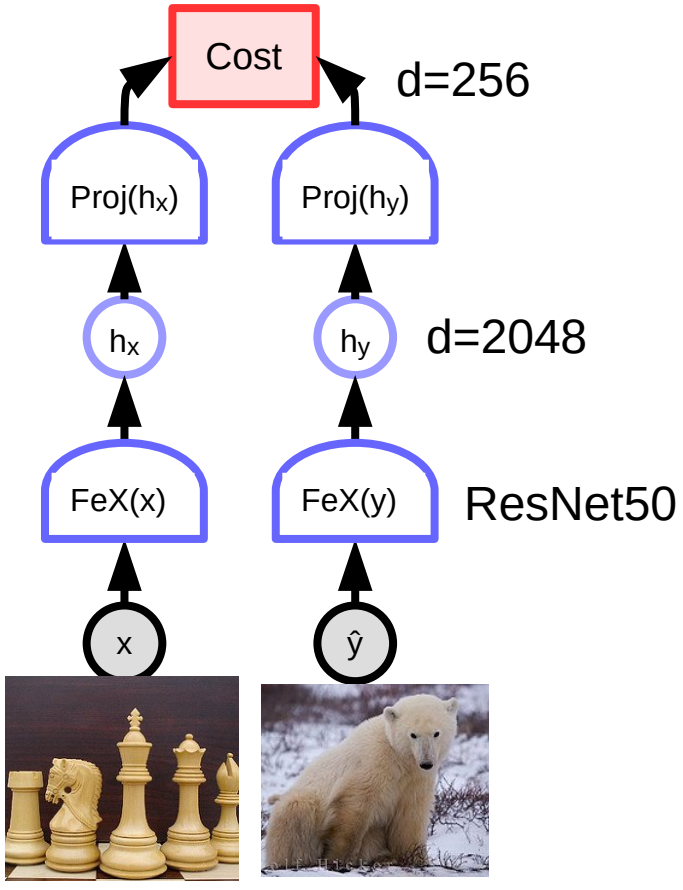
Negative pair:
Make F large

Contrastive Joint Embedding

Make F(x,y) small



Make F(x,ŷ) large

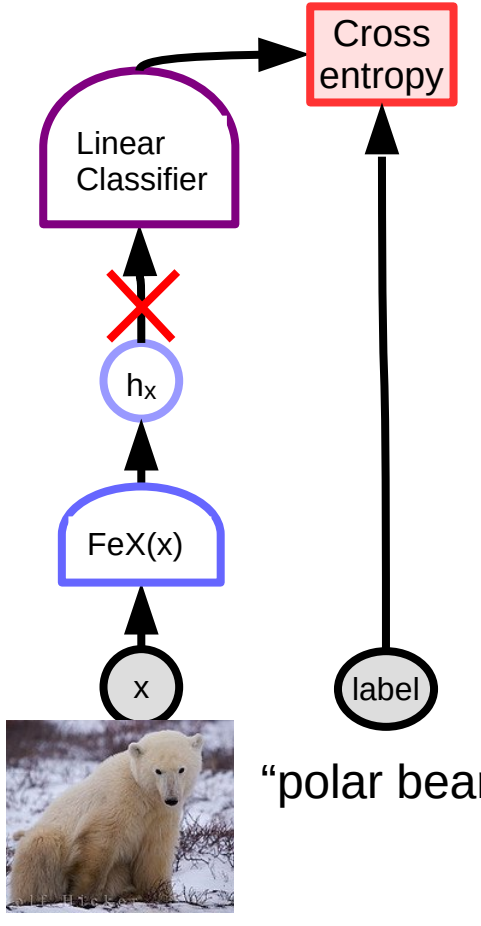


d=256

d=2048

ResNet50

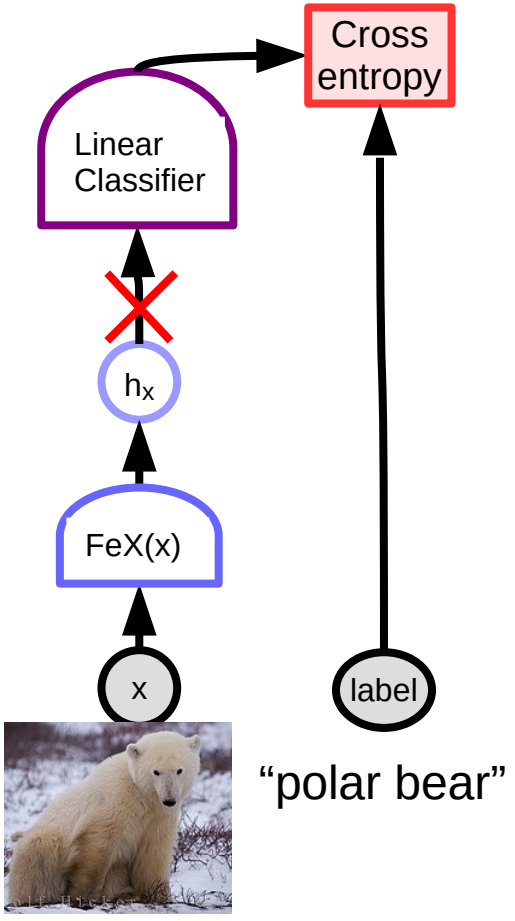
Training a supervised linear head



“polar bear”

Contrastive Joint Embedding

- **Issues:**
  - Hard negative mining
  - Expensive computationally
  - Only works for small dimension of embeddings (256)
- **Successful examples for image recognition:**
  - PIRL [Misra et al. Arxiv:1912.01991]
  - MoCo [He et al. Arxiv:1911.05722]
  - SimCLR [Chen et al. Arxiv:2002.05709]
- **Use InfoNCE group loss**



$$\mathcal{L}(x, y, \hat{y_1}, \ldots \hat{y}_q, w) = F_w(x, y) + \log \left[e^{-F_w(x,y)} + \sum_{i=1}^{q} e^{-F_w(x,\hat{y}_i,w)} \right]$$

Contrastive Methods: group losses

- **Push down on a group of data points, push up on a group of contrastive points**
  - General group loss on p^+ data points and p^- contrastive points:

$$\mathcal{L}(x_1 \dots x_{p^+}, y_1 \dots y_{p^+}, \hat{y}_1 \dots \hat{y}_{p^-}, w) = H\left(F(x_1, y_1), \dots F(x_{p^+}, y_{p^+}), F(x_1, \hat{y_1}), \dots F(x_{p^+}, \hat{y_{p^+}}), M(Y_{1\dots p^+}, \hat{Y}_{1\dots p^-})\right)$$

    - Where H must be an increasing fn of the data energies and decreasing fn of the contrastive point energies within the margin.
    - M is a margin matrix for all pairs of y and ŷ in the group.
  - Example: Neighborhood Component Analysis, Noise Contrastive Estimation, InfoNCE (implicit infinite margin) [Goldberger 2005] [Gutmann 2010]... [Misra 2019] [Chen 2020]

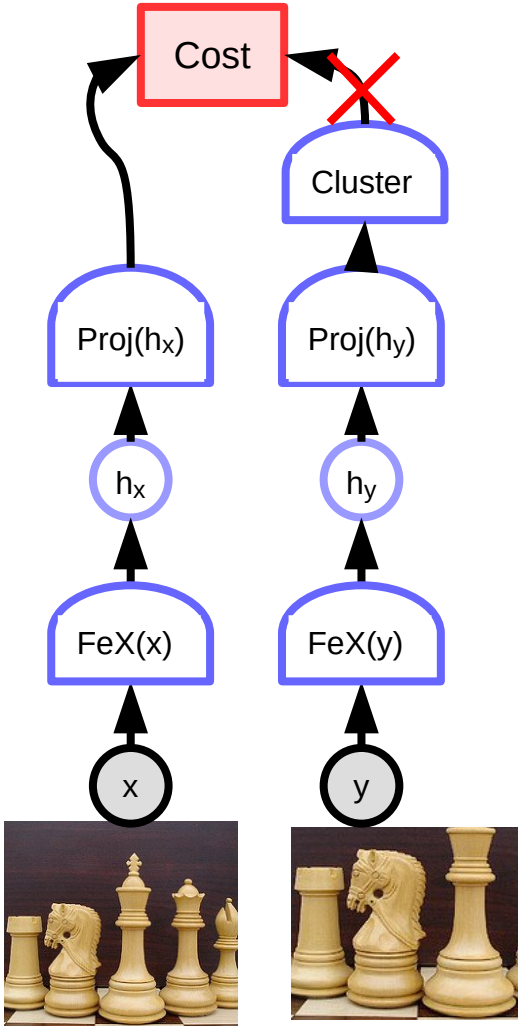
$$\mathcal{L}(x, y, \hat{y}_1, \dots \hat{y}_{p^-}, w) = -\log \frac{e^{-F_w(x,y)}}{e^{-F_w(x,y)} + \sum_{i=1}^{p^-} e^{-F_w(x,\hat{y}_i,w)}}$$

Loss function zoo for contrastive EBM training

	Method	Energy	$\hat{y}$ Generation	Loss
1	Max Likelihood	discrete y	exhaustive	$F_w(x,y) + \log \sum_{y' \in \mathcal{Y}} \exp(-F_w(x,y'))$
2	Max Likelihood	tractable	exhaustive	$F_w(x,y) + \log \int_{y' \in \mathcal{Y}} \exp(-F_w(x,y'))$
3	Max likelihood	any	MC or MCMC	$F_w(x,y) - F_w(x,\hat{y})$
4	Contr. Divergence	any	trunc'd MCMC	$F_w(x,y) - F_w(x,\hat{y})$
5	Pairwise Hinge	any	most offending	$[F_w(x,y) - F_w(x,\hat{y}) + m(y,\hat{y})]^+$
6	Min-Hinge	positive	most offending	$F_w(x,y) + [m(y,\hat{y}) - F_w(x,\hat{y})]^+$
6	Square-Hinge	divergence	most offending	$F_w(x,y)^2 + \left([m(y,\hat{y}) - F_w(x,\hat{y})]^+\right)^2$
7	Square-Exp	any	most offending	$F_w(x,y)^2 + \exp(-\beta F_w(x,\hat{y}))$
8	Logistic	any	most offending	$\log(1 + \exp(F_w(x,y) - F_w(x,\hat{y}))$
9	GAN	any	$\hat{y} = g_u(z)$	$H(F_w(x,y), F_w(x,\hat{y}), m(y,\hat{y}))$
10	Denoising AE	$D(y, g_w(y))$	$\hat{y} = N(y)$	$D(y, g_w(\hat{y})$

Quantization Methods

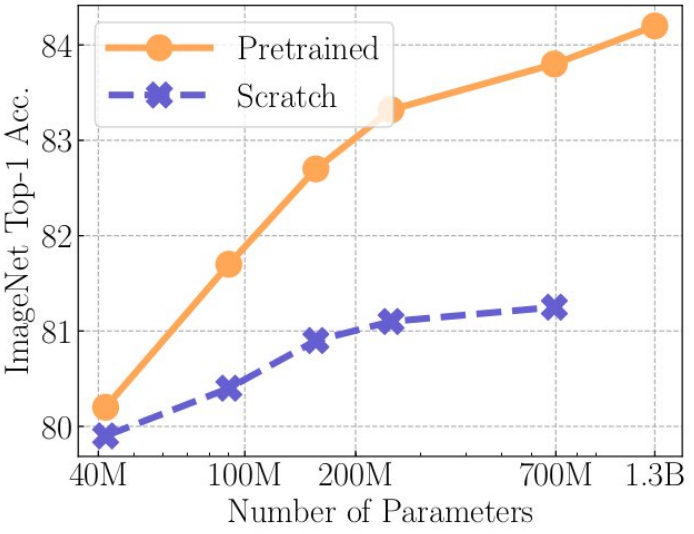
- **K-means clustering on embedding vectors**
  - Ensuring that all clusters are populated
    - Sinkhorn-Knapp procedure (information maximization)
  - Cluster centers used as targets for student branch
- **Examples**
  - DeepCluster [Caron arXiv:1807.05520]
  - SwAV [Caron arXiv:2006.09882]
- **Advantage:**
  - Works really well!
  - Uses large distortions (multicrop)
  - Scales to very large datasets

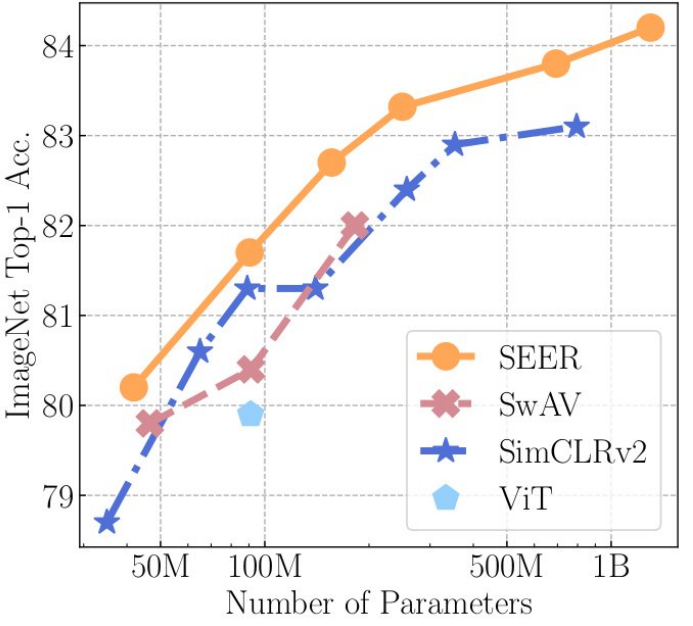


SEER [Goyal et al. ArXiv:2103.01988]

- **SwAV training on 1 billion random IG images**
- **RegNet architecture**
- **Fine-tuned on various datasets**
  - ImgNet full: 84.% top-1 correct
  - ImgNet 10%: 77.9%, ImgNet 1%: 60.5%
  - Inaturalist: 50.8%, Places205: 62.7%
  - Pascal VOC2007: 92.6%
- **Code: https://vissl.ai/**

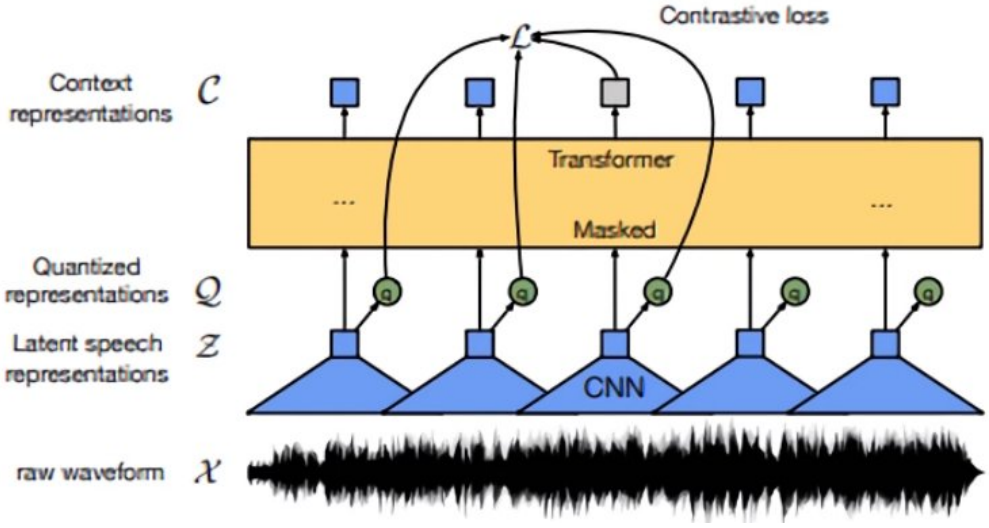
Method	Data	#images	Arch.	#param.	Top-1
DeeperCluster [6]	YFCC100M	96M	VGG16	138M	74.9
ViT [14]	JFT	300M	ViT-B/16	91M	79.9
SwAV [7]	IG	1B	RX101-32x16d	182M	82.0
SimCLRv2 [9]	ImageNet	1.2M	RN152w3+SK	795M	83.1
SEER	IG	1B	RG128	693M	83.8
SEER	IG	1B	RG256	1.3B	**84.2**

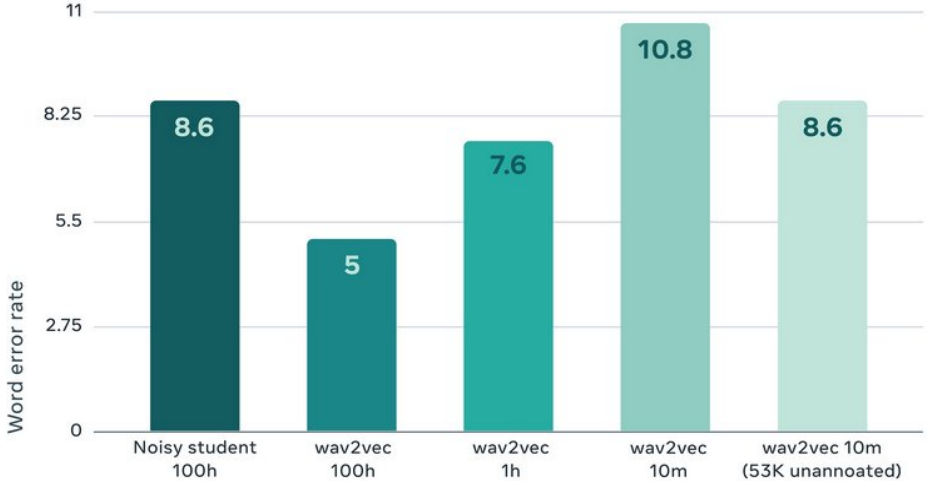




Wav2Vec 2.0: SSL for speech recognition

- **Pre-train on 960h of unlabeled speech,**
- **then train with 10 minutes, 1h or 100h of labeled speech**
- **Results on LibriSpeech**
  - Wav2vec on 10 minutes = Same WER as previous SOTA on 100h
  - Papers: [Baevski et al. NeurIPS 2020] [Xu et al. ArXiv:2010.11430]
  - Code: Github: PyTorch/fairseq

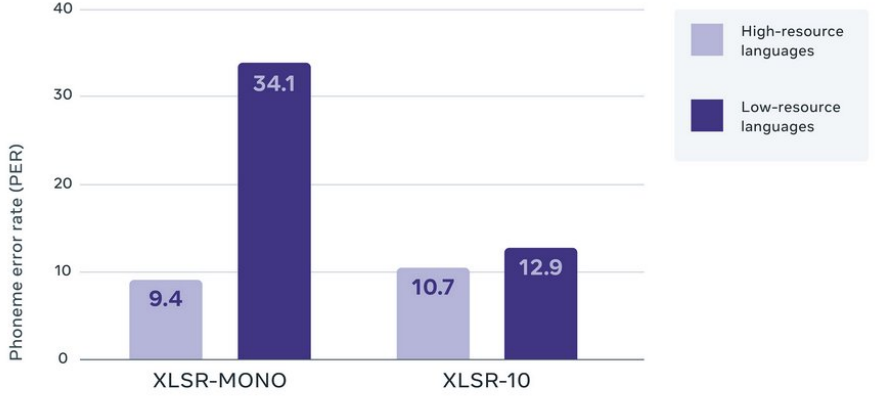




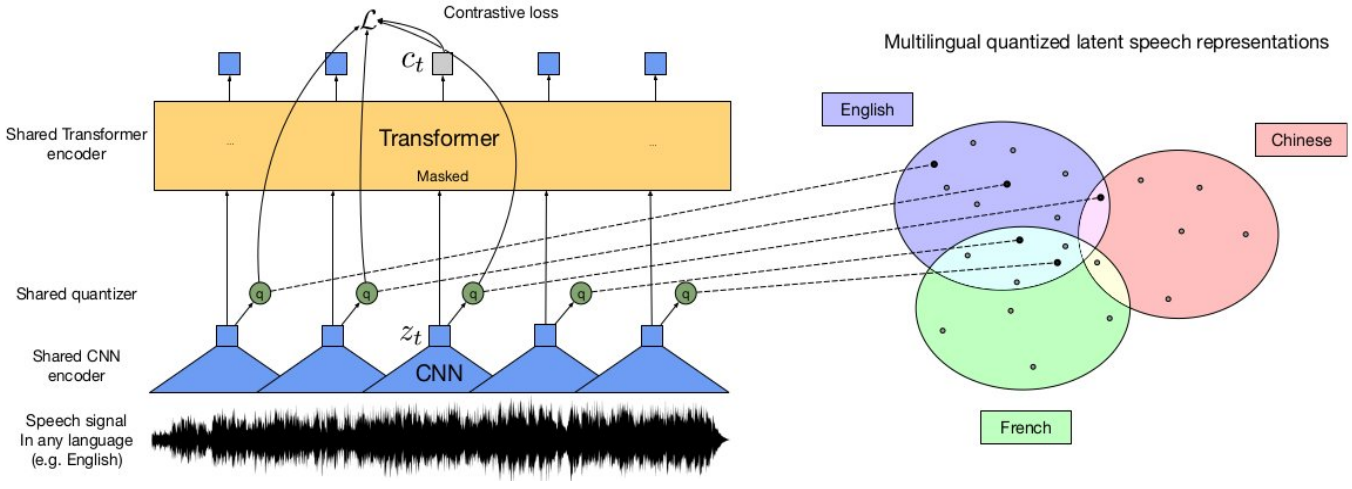
WER for Noisy Student self-training with 100 hours of labeled data. Wav2vec 2.0 with 100 hours, 1 hour, and only 10 minutes of labeled data. All models use the remainder of the LibriSpeech corpus (total 960 hours) as unannotated data, except for the last result, which uses 53K hours from LibriVox.

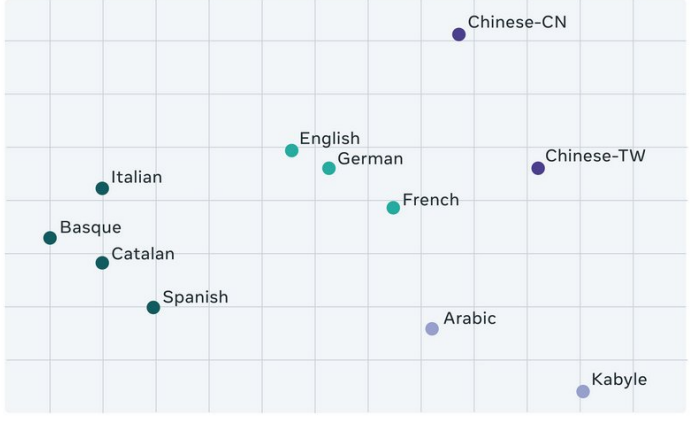
XLSR: multilingual speech recognition

- **Multilingual self-supervised ASR**
  - [Conneau arXiv:2006.13979]
  - Raw audio → ConvNet → Transformer
  - CommonVoice: 72% reduction of PER
  - BABEL: 16% reduction of WER



Results on the Common Voice benchmark in terms of phoneme error rate (PER), comparing training on each language individually (XLSR-Mono) with training on all 10 languages simultaneously (XLSR-10).

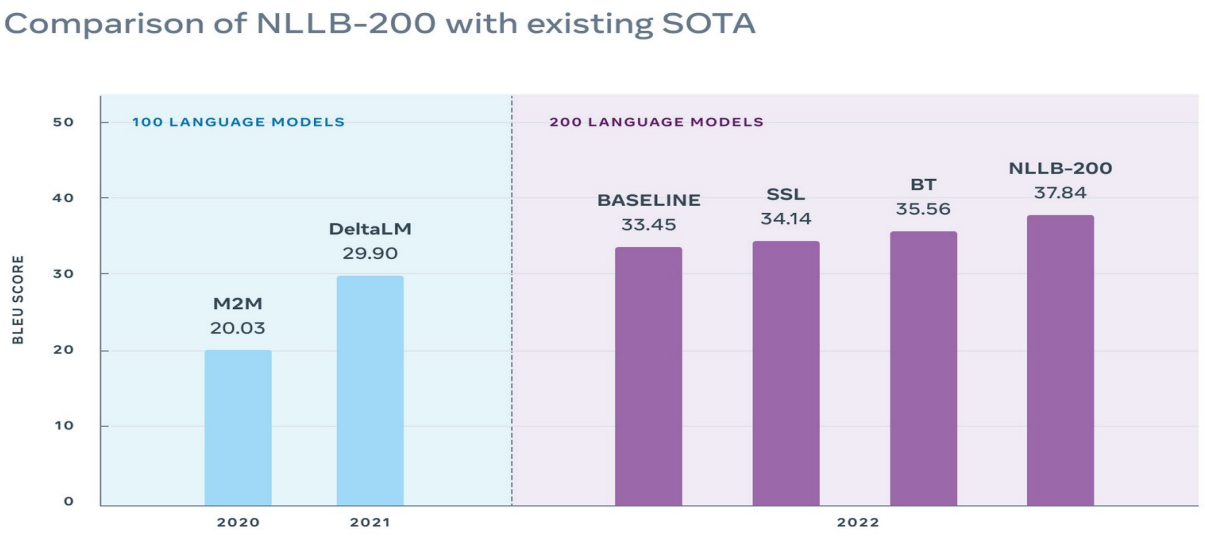




Visualization of how the learned units are used across languages. Graph shows a 2D PCA plot of how units are used in each language. Languages closer to each other, like English and German or Basque and Catalan, tend to use similar units.

No Language Left Behind (NLLB)

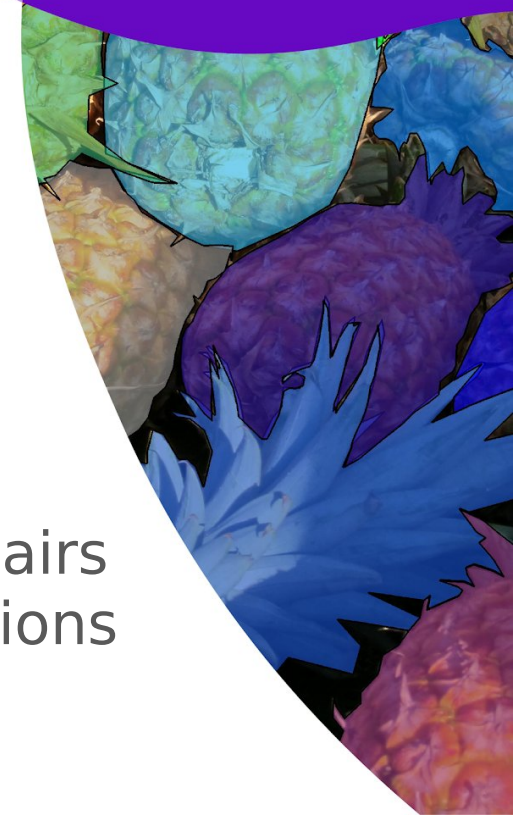
- **Language translation between 202 languages**
  - in any of the 40602 directions
  - Training set: 18 billion pairs of sentences for 2440 language directions
  - Most pairs have less than 1 million sentences
  - https://ai.facebook.com/research/no-language-left-behind/
- **A single neural net with 54 billion parameters**
- **Performance gets better as more languages are added**
- **Relies on Self-Supervised Learning and back-translation.**



No Language Left Behind (NLLB)

Acehnese
Acehnese
Mesopotamian Arabic
Ta'izzi-Adeni Arabic
Tunisian Arabic
Afrikaans
South Levantine Arabic
Akan
Amharic
North Levantine Arabic
Modern Standard Arabic
Modern Standard Arabic
Najdi Arabic
Moroccan Arabic
Egyptian Arabic
Assamese
Asturian
Awadhi
Central Aymara
South Azerbaijani
North Azerbaijani
Bashkir
Bambara
Balinese
Belarusian
Bemba
Bengali
Bhojpuri
Banjar
Banjar
Standard Tibetan
Bosnian
Buginese
Bulgarian
Catalan
Cebuano
Czech
Chokwe
Central Kurdish
Crimean Tatar
Welsh
Danish
German
Southwestern Dinka
Dyula
Dzongkha
Greek
English
Esperanto
Estonian
Basque
Ewe
Faroese
Fijian
Finnish
Fon
French
Friulian
Nigerian Fulfulde
Scottish Gaelic
Irish
Galician
Guarani
Gujarati
Haitian Creole
Hausa
Hebrew
Hindi
Chhattisgarhi
Croatian
Hungarian
Armenian
Igbo
Ilocano
Indonesian
Icelandic
Italian
Javanese
Japanese
Kabyle
Jingpho
Kamba
Kannada
Kashmiri
Kashmiri
Georgian
Central Kanuri
Central Kanuri
Kazakh
Kabiyè
Kabuverdianu
Khmer
Kikuyu
Kinyarwanda
Kyrgyz
Kimbundu
Northern Kurdish
Kikongo
Korean
Lao
Ligurian
Limburgish
Lingala
Lithuanian
Lombard
Latgalian
Luxembourgish
Luba-Kasai
Ganda
Luo
Mizo
Standard Latvian
Magahi
Maithili
Malayalam
Marathi
Minangkabau
Minangkabau
Macedonian
Plateau Malagasy
Maltese
Meitei
Halh Mongolian
Mossi
Maori
Burmese
Dutch
Norwegian Nynorsk
Norwegian Bokmål
Nepali
Northern Sotho
Nuer
Nyanja
Occitan
West Central Oromo
Odia
Pangasinan
Eastern Panjabi
Papiamento
Western Persian
Polish
Portuguese
Dari
Southern Pashto
Ayacucho Quechua
Romanian
Rundi
Russian
Sango
Sanskrit
Santali
Sicilian
Shan
Sinhala
Slovak
Slovenian
Samoan
Shona
Sindhi
Somali
Southern Sotho
Spanish
Tosk Albanian
Sardinian
Serbian
Swati
Sundanese
Swedish
Swahili
Silesian
Tamil
Tatar
Telugu
Tajik
Tagalog
Thai
Tigrinya
Tamasheq
Tamasheq
Tok Pisin
Tswana
Tsonga
Turkmen
Tumbuka
Turkish
Twi
Central Atlas Tamazight
Uyghur
Ukrainian
Umbundu
Urdu
Northern Uzbek
Venetian
Vietnamese
Waray
Wolof
Xhosa
Eastern Yiddish
Yoruba
Yue Chinese
Chinese
Chinese
Standard Malay
Zulu

Regularized Methods
for joint embedding architectures

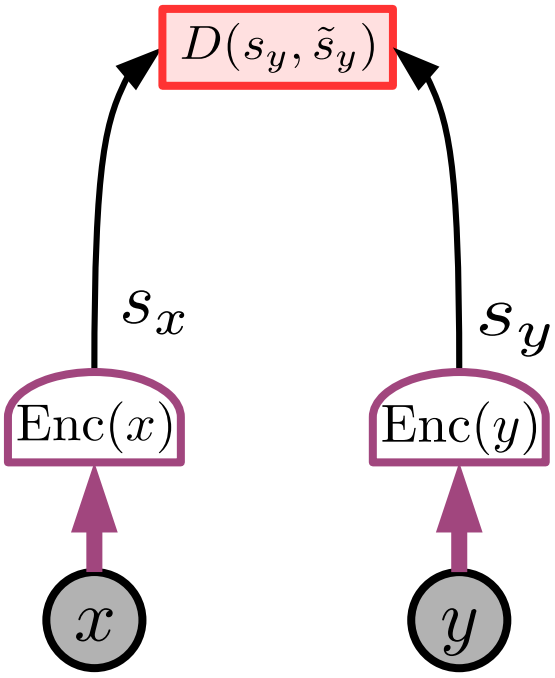


This is the cool stuff!

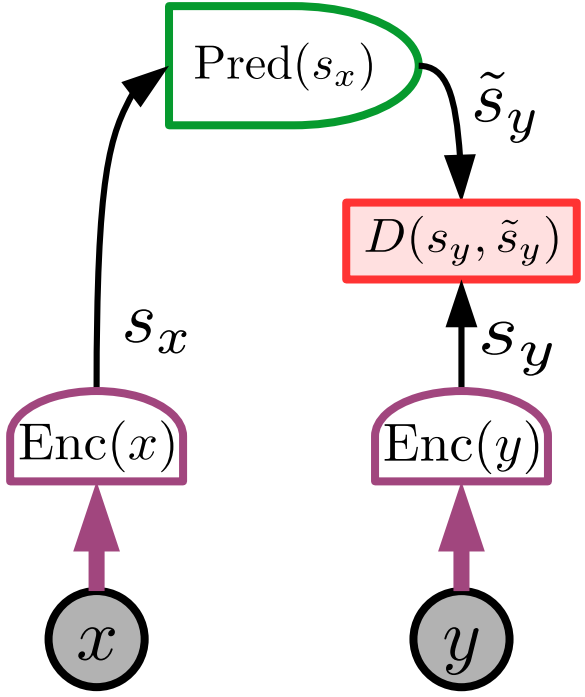
Push down on the energy of compatible sample pairs
Maximize the information capacity of representations

Joint Embedding Architecture (JEA)

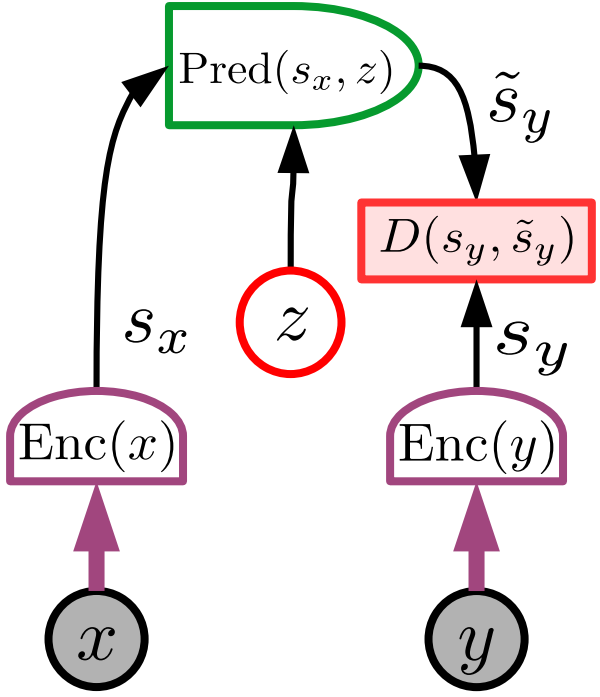
- **Computes abstract representations for x and y**
- **Makes predictions in representation space**



a) Joint Embedding Architecture (JEA)



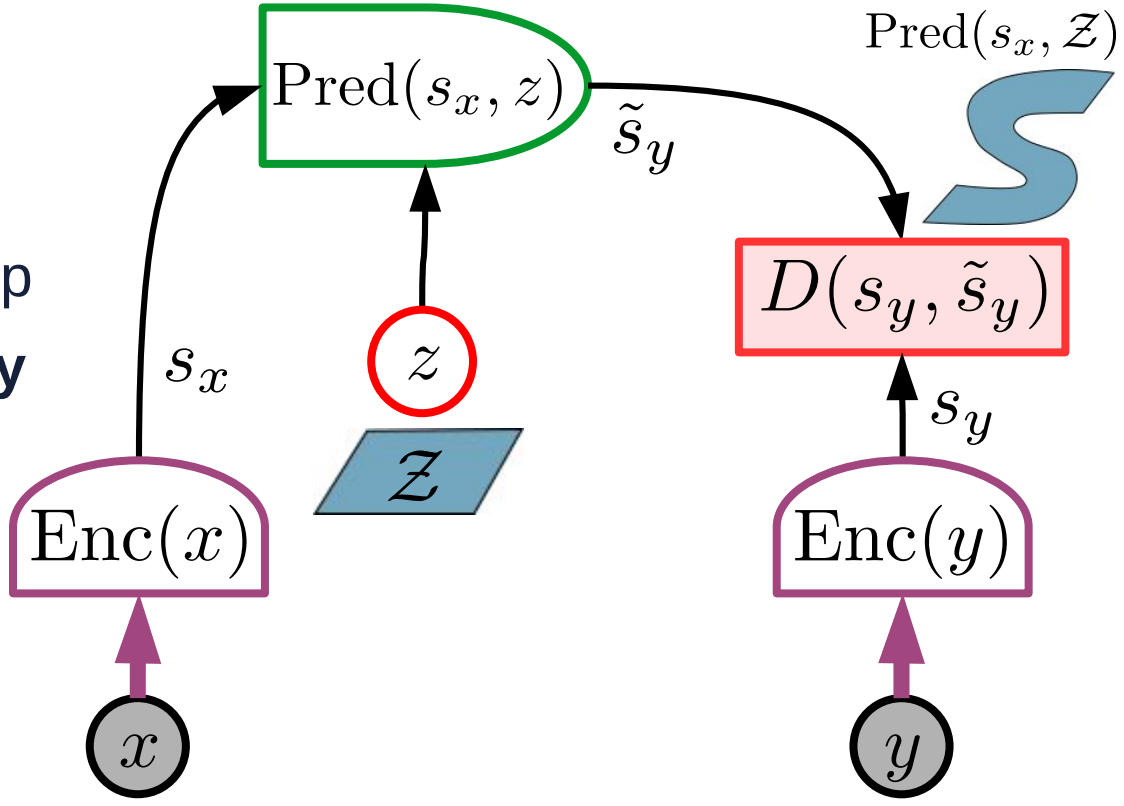
b) Deterministic Joint Embedding Predictive Architecture (DJEPA)



c) Joint Embedding Predictive Architecture (JEPA)

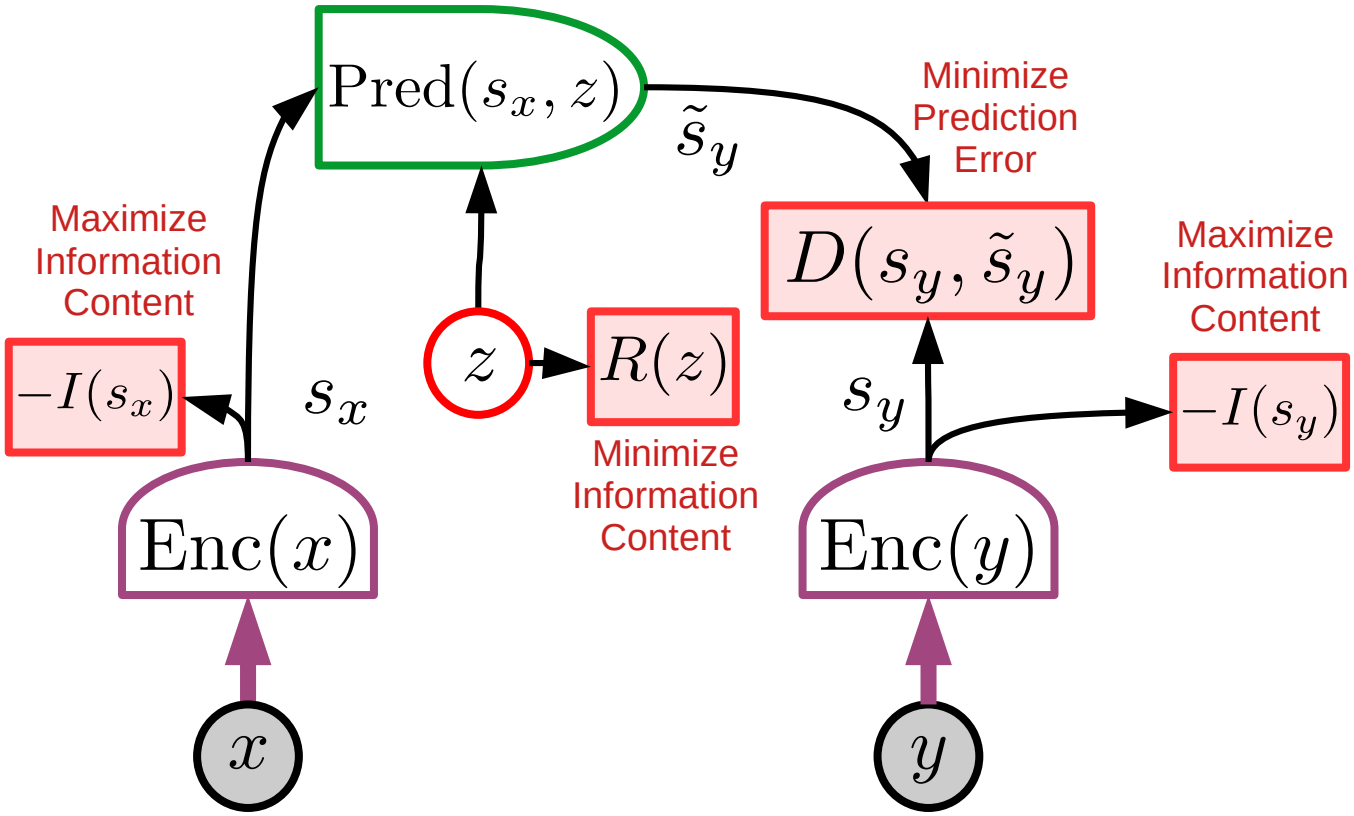
Joint Embedding Predictive Architecture (JEPA)

- **Computes abstract representations for x and y**
- **Makes predictions in representation space**
  - Can use a latent variable to help
- **Does not need to predict every details of y**
  - Enc(y) can eliminate irrelevant details through invariances
- **Tries to make the representations predictable from each other.**

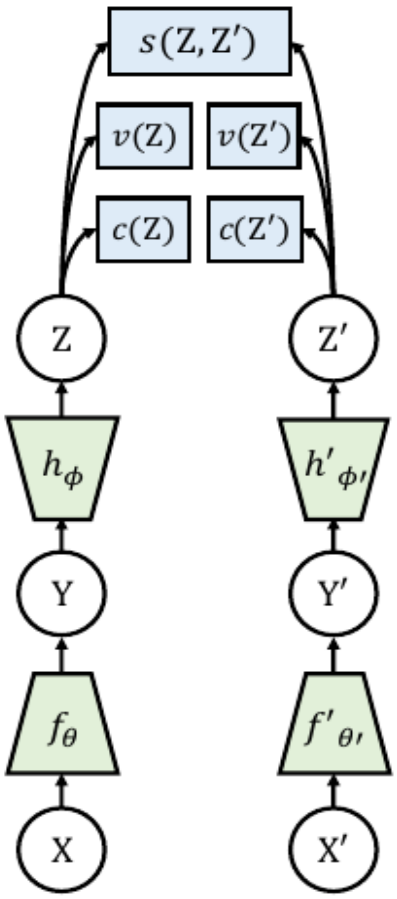


Training a JEPA (non contrastively)

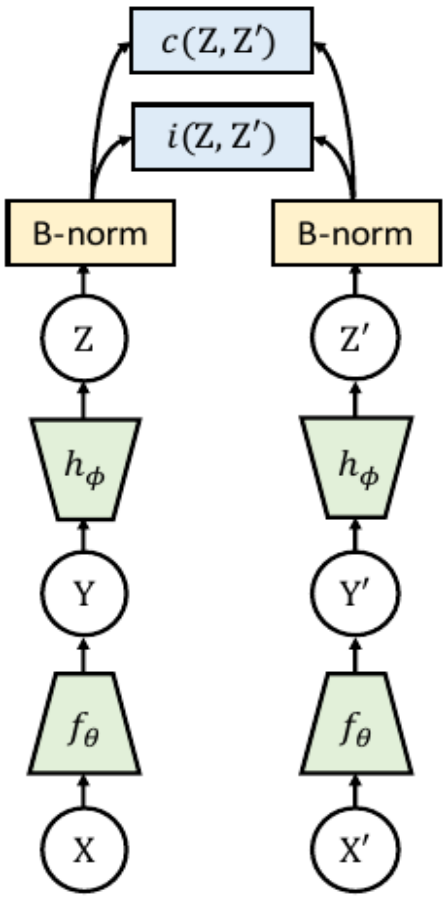
- **Four terms in the cost**
  - Maximize information content in representation of x
  - Maximize information content in representation of y
  - Minimize Prediction error
  - Minimize information content of latent variable z



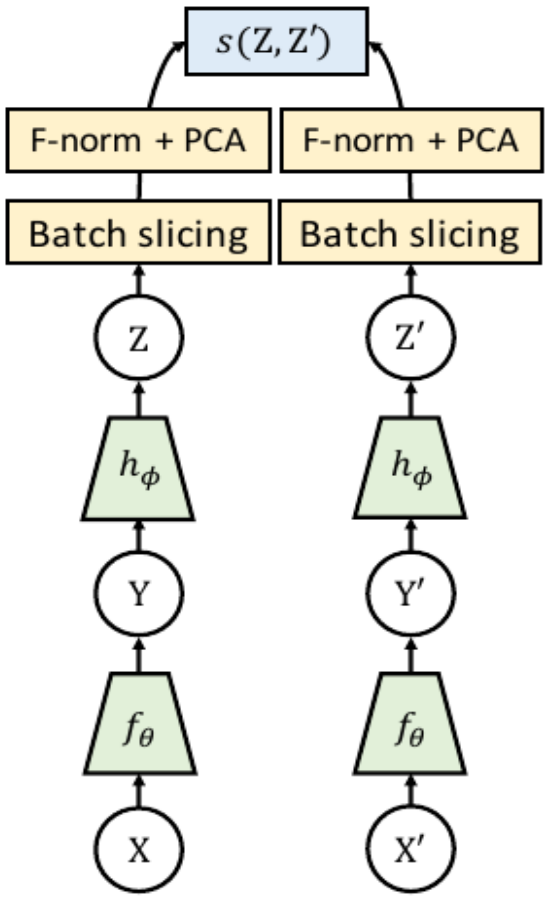
Regularized (Non-Contrastive) Joint Embedding Methods



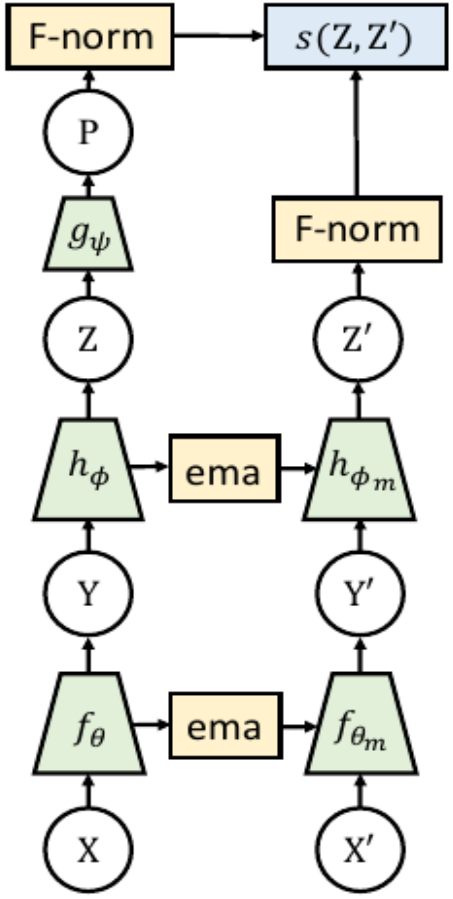
(a) VICReg



(b) Barlow Twins



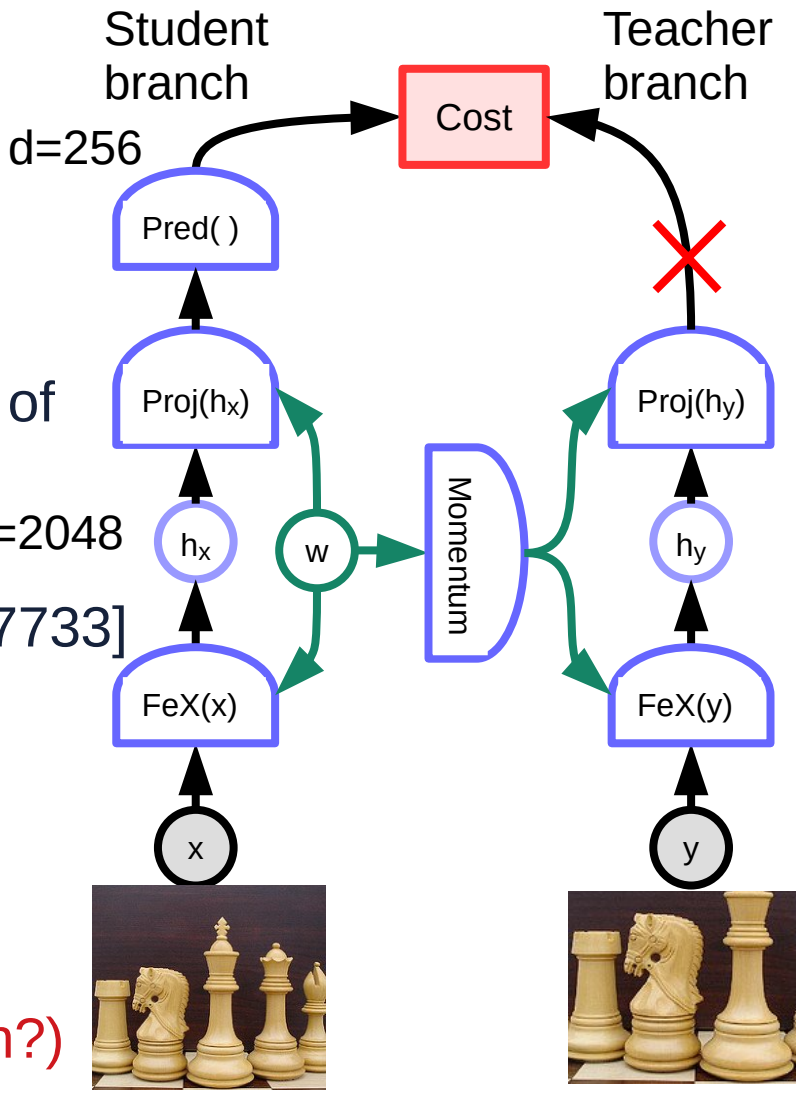
(c) W-MSE



(d) BYOL

Distillation Methods

- **Modified Siamese nets**
  - Predictor head eliminates variation of representations due to distortions
  - BYOL: Teacher branch uses a moving-average of the parameters of the student branch
- **Examples:**
  - Bootstrap Your Own Latents [Grill arXiv:2006.07733]
  - SimSiam [Chen & He arXiv:2011.10566]
- **Advantages**
  - No negative samples
- **Issues**
  - Not clear why they don't collapse (normalization?)

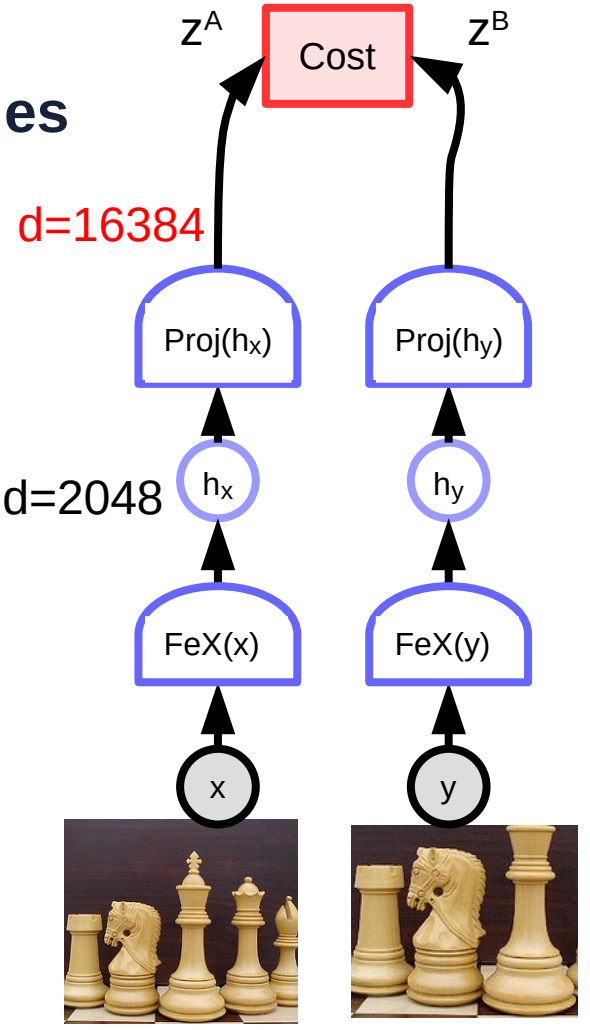


Information Maximization

- **Minimizes redundancy between embedding variables**
  - Maximizes information content of embedding vectors
- **Example: Barlow Twins**
  - [Zbontar et al. ArXiv:2103.03230]
  - Maximizes normalized correlation between the same variable in the two branches over a batch.
  - Minimizes normalized correlation between different variables in the two branches
  - Centered vectors z^A,z^B

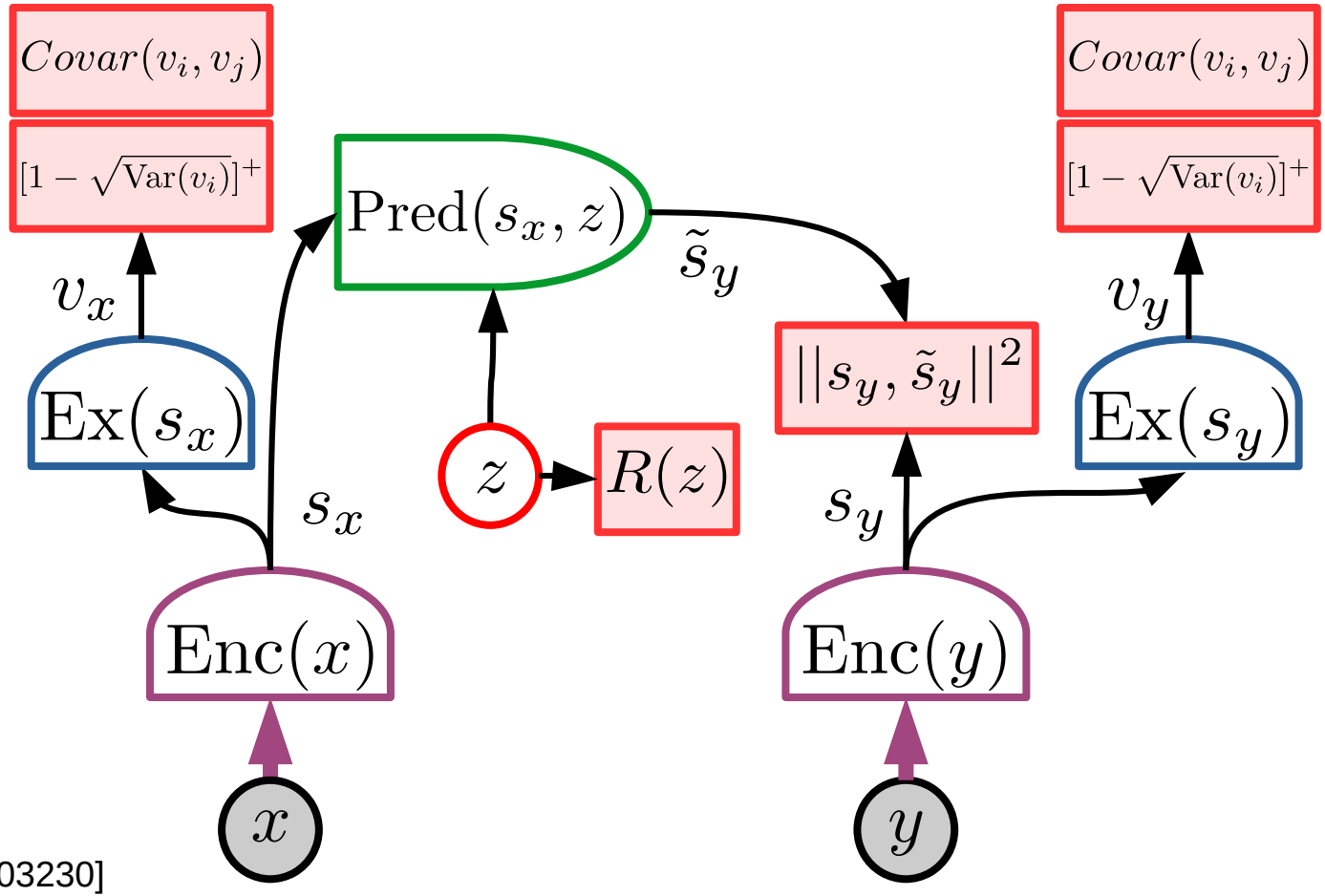
$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b \left(z^A_{b,i}\right)^2} \sqrt{\sum_b \left(z^B_{b,j}\right)^2}}$$

$$\mathcal{L}_{\mathcal{BT}} \triangleq \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invariance term}} + \lambda \underbrace{\sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2}_{\text{redundancy reduction term}}$$



VICReg: Variance, Invariance, Covariance Regularization

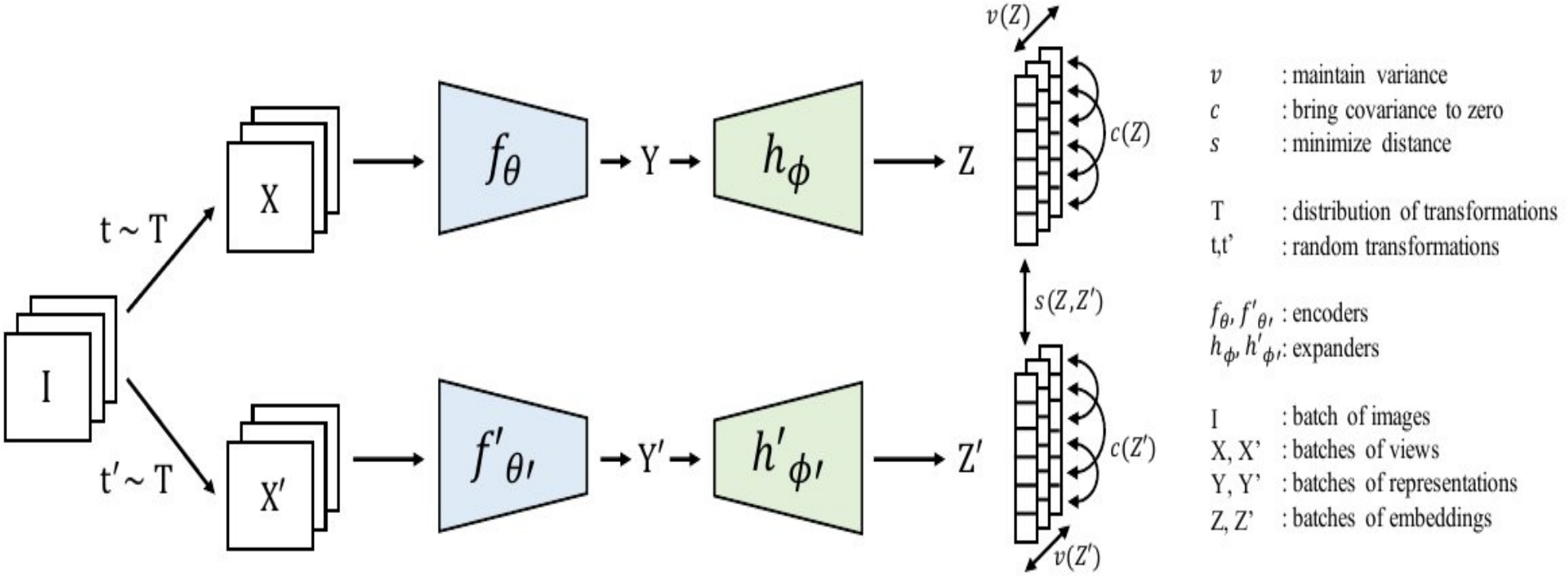
- **Variance:**
  - Maintains variance of components of representations
- **Covariance:**
  - Decorrelates components of covariance matrix of representations
- **Invariance:**
  - Minimizes prediction error.



Barlow Twins [Zbontar et al. ArXiv:2103.03230]
VICReg [Bardes, Ponce, LeCun arXiv:2105.04906, ICLR 2022]

VICReg: Variance-Invariance-Covariance Regularization

- VICReg: [Bardes, Ponce, LeCun arXiv:2105.04906, ICLR 2022]
- Improvement on Barlow Twins [Zbontar et al. ArXiv:2103.03230]
- Maximizes a measure of mutual information between the two outputs



VICReg: Results with linear head and semi-supervised.

Method	Linear		Semi-supervised			
	Top-1	Top-5	Top-1		Top-5	
			1%	10%	1%	10%
Supervised	76.5	-	25.4	56.4	48.4	80.4
MoCo He et al. (2020)	60.6	-	-	-	-	-
PIRL Misra & Maaten (2020)	63.6	-	-	-	57.2	83.8
CPC v2 Hénaff et al. (2019)	63.8	-	-	-	-	-
CMC Tian et al. (2019)	66.2	-	-	-	-	-
SimCLR Chen et al. (2020a)	69.3	89.0	48.3	65.6	75.5	87.8
MoCo v2 Chen et al. (2020c)	71.1	-	-	-	-	-
SimSiam Chen & He (2020)	71.3	-	-	-	-	-
SwAV Caron et al. (2020)	71.8	-	-	-	-	-
InfoMin Aug Tian et al. (2020)	73.0	91.1	-	-	-	-
OBoW Gidaris et al. (2021)	73.8	-	-	-	82.9	90.7
BYOL Grill et al. (2020)	74.3	91.6	53.2	68.8	78.4	89.0
SwAV (w/ multi-crop) Caron et al. (2020)	75.3	-	53.9	70.2	78.5	89.9
Barlow Twins Zbontar et al. (2021)	73.2	91.0	55.0	69.7	79.2	89.3
VICReg (ours)	73.2	91.1	54.8	69.5	79.4	89.5

VICReg: Results with transfer tasks.

Method	Linear Classification			Object Detection		
	Places205	VOC07	iNat18	VOC07+12	COCO det	COCO seg
Supervised	53.2	87.5	46.7	81.3	39.0	35.4
MoCo He et al. (2020)	46.9	79.8	31.5	-	-	-
PIRL Misra & Maaten (2020)	49.8	81.1	34.1	-	-	-
SimCLR Chen et al. (2020a)	52.5	85.5	37.2	-	-	-
MoCo v2 Chen et al. (2020c)	51.8	86.4	38.6	82.5	39.8	36.1
SimSiam Chen & He (2020)	-	-	-	82.4	-	-
BYOL Grill et al. (2020)	54.0	86.6	47.6	-	40.4†	37.0†
SwAV (m-c) Caron et al. (2020)	56.7	88.9	48.6	82.6	41.6	37.8
OBoW Gidaris et al. (2021)	56.8	89.3	-	82.9	-	-
Barlow Twins Grill et al. (2020)	54.1	86.2	46.5	82.6	40.0†	36.7†
VICReg (ours)	54.3	86.6	47.0	82.4	39.4	36.4

VICReg: no need for normalization, momentum encoder, predictor...

Table 3: **Effect of incorporating variance and covariance regularization in different methods.** Top-1 ImageNet accuracy with the linear evaluation protocol after 100 pretraining epochs. For all methods, pretraining follows the architecture, the optimization and the data augmentation protocol of the original method using our reimplementation. ME: Momentum Encoder. SG: stop-gradient. PR: predictor. BN: Batch normalization layers after input and inner linear layers in the expander. No Reg: No additional regularization. Var Reg: Variance regularization. Var/Cov Reg: Variance and Covariance regularization. Unmodified original setups are marked by a †.

Method	ME	SG	PR	BN	No Reg	Var Reg	Var/Cov Reg
BYOL	✓	✓	✓	✓	69.3†	70.2	69.5
SimSiam		✓	✓	✓	67.9†	68.1	67.6
SimSiam		✓	✓		35.1	67.3	67.1
SimSiam		✓			collapse	56.8	66.1
VICReg			✓		collapse	56.2	67.3
VICReg			✓	✓	collapse	57.1	68.7
VICReg				✓	collapse	57.5	68.6†
VICReg					collapse	56.5	67.4

VICReg: Variance/Covariance regularization helps other methods

Table 5: **Impact of variance-covariance regularization.** Inv: a invariance loss is used, $\lambda > 0$, Var: variance regularization, $\mu > 0$, Cov: covariance regularization, $\nu > 0$, in Eq. (6).

Method	λ	μ	ν	Top-1
Inv	1	0	0	collapse
Inv + Cov	25	0	1	collapse
Inv + Cov	0	25	1	collapse
Inv + Var	1	1	0	57.5
Inv + Var + Cov (VICReg)	25	25	1	68.6

Table 6: **Impact of normalization.** Std: variables are centered and divided by their standard deviation over the batch. This is applied or not to the embedding and the expander hidden layers. l_2: the embedding vectors are l_2-normalized.

Expander	Embedding	Top-1
Std	None	68.6
Std	Std	68.4
None	None	67.4
None	Std	67.2
Std	l_2	65.1

VICReg: No need for weight sharing between the branches!

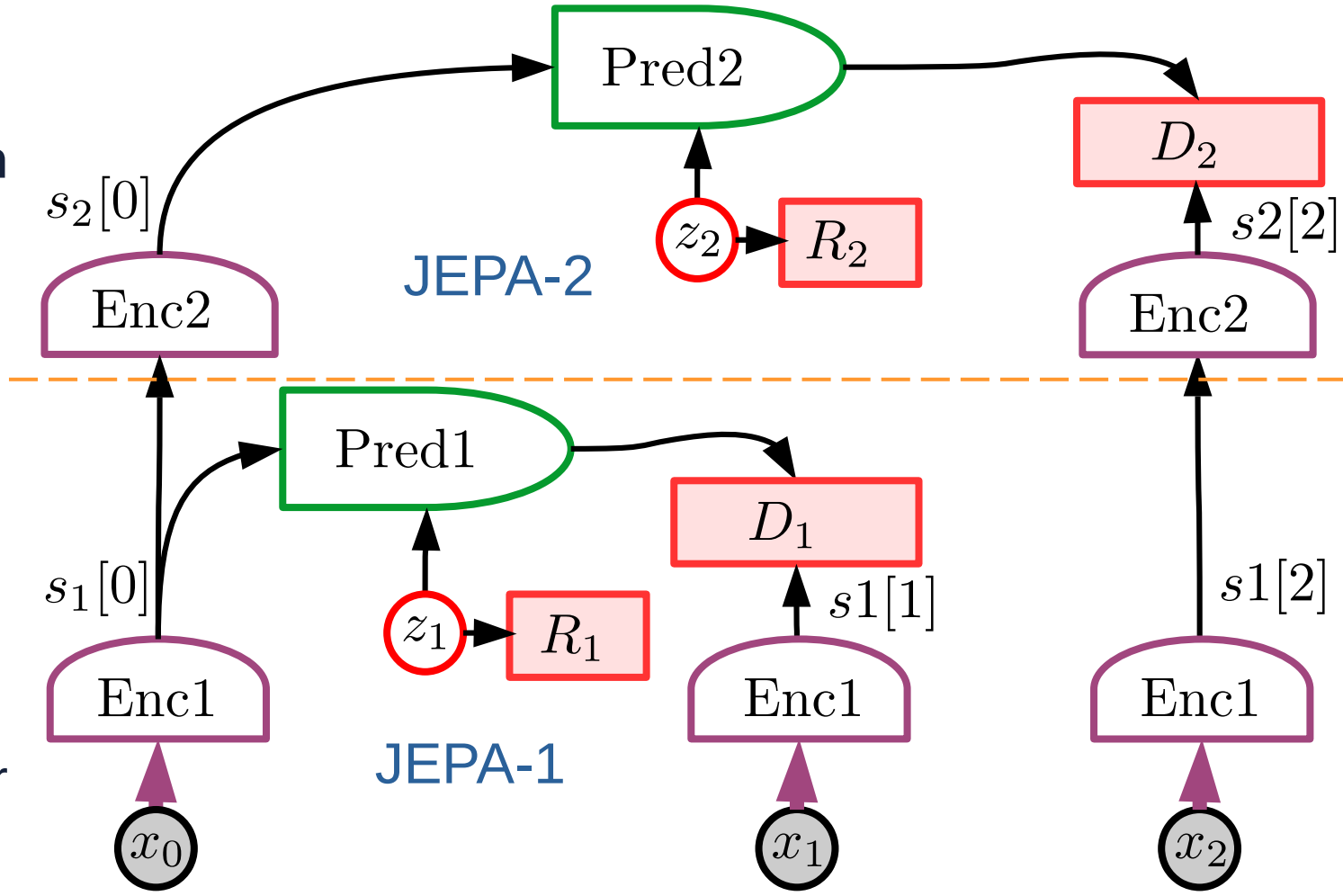
- **No need for weight sharing!**
- **The two branches can take inputs of different nature.**
- **Opens the door to many applications of non-contrastive SSL to many domains**

Table 4: **Impact of sharing weights or not between branches.** Top-1 accuracy on linear classification with 100 pretraining epochs. In all settings, the encoder and expander of both branches share the same architecture, but either share weights (✓), or have different weights in the two branches.

Encoder	Expander	Top-1
		66.5
	✓	67.3
✓		67.8
✓	✓	68.6

Hierarchical JEPA: Multi-level, multi-timescale Predictions

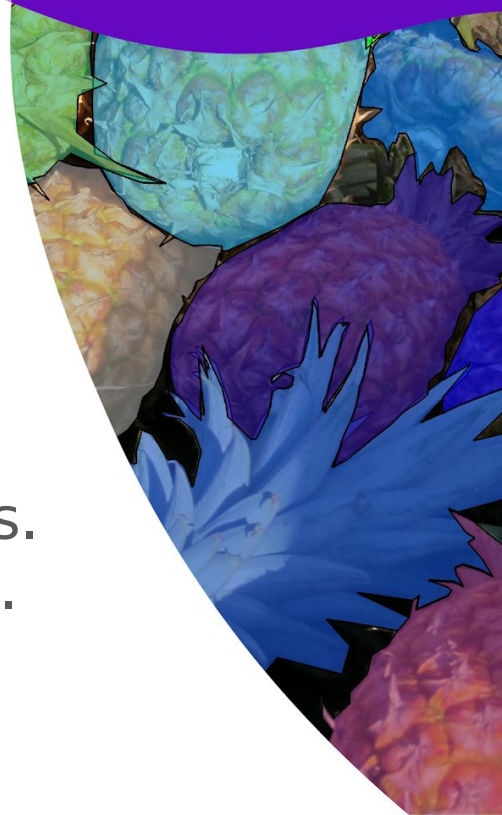
- **Low-level representations can only predict in the short term.**
  - Too much details
  - Prediction is hard
- **Higher-level representations can predict in the longer term.**
  - Less details.
  - Prediction is easier



JEPA and H-JEPA

- **Are <u>not</u> generative models**
  - Because prediction takes place in representation space
- **Are <u>not</u> probabilistic models**
  - Because the encoder of y is not invertible
  - There is no simple way get a JEPA to produce a normalized distribution p(y|x), unless the y encoder is invertible (but then, what's the point?)
- **H-JEPA is hierarchical**
  - Latent variables are fed to the next layer

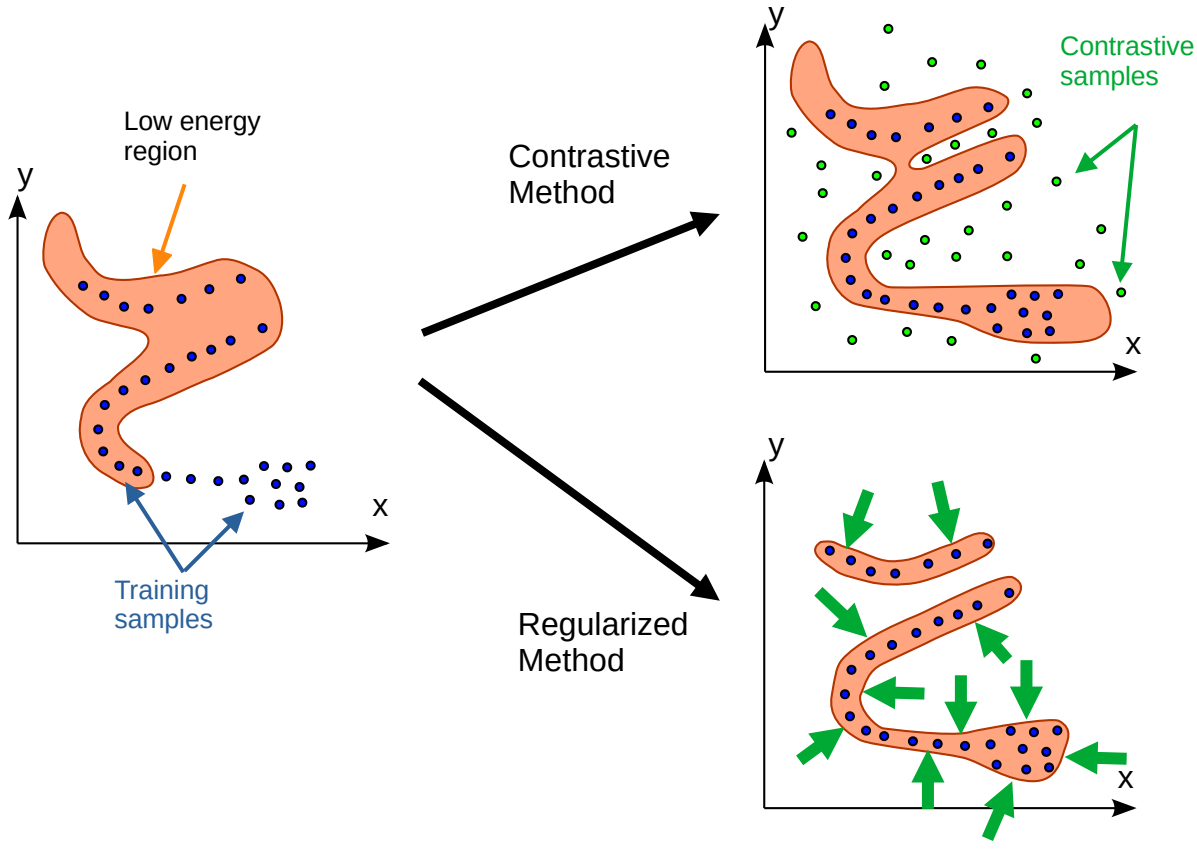
Regularized Methods
for latent-variable architectures



Push down on the energy of training samples.
Minimize the capacity of the latent variables.

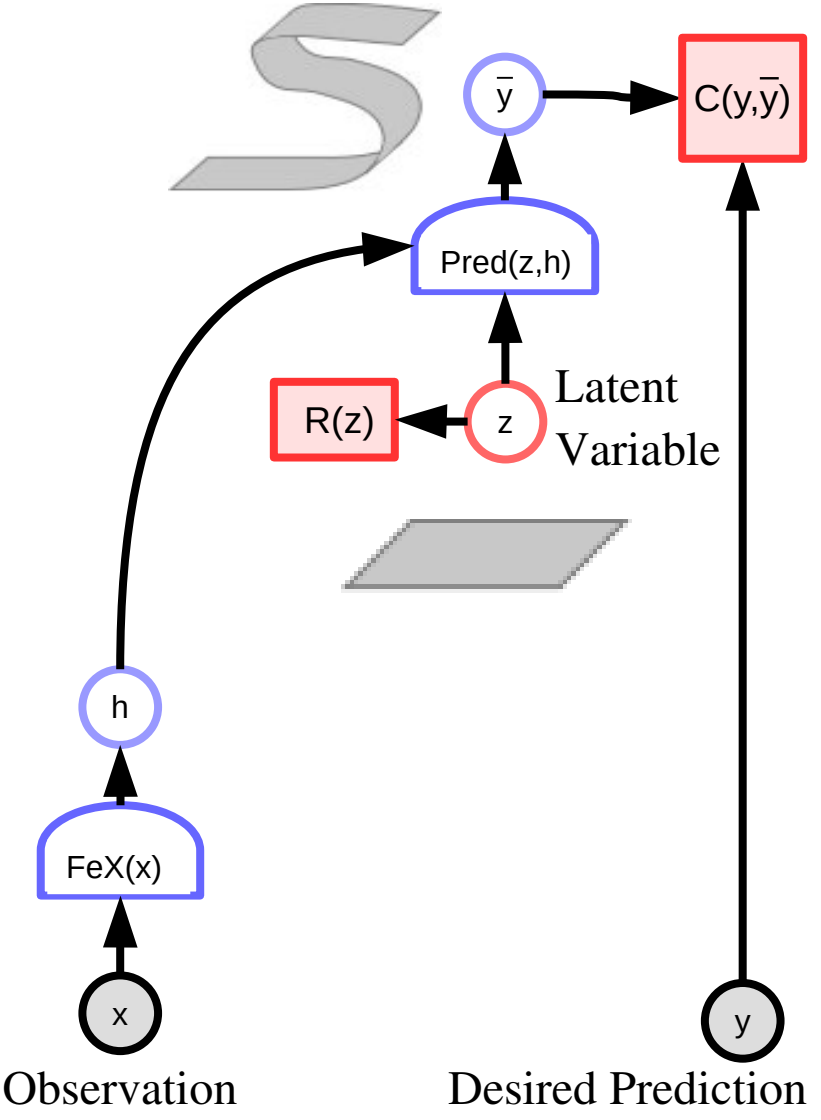
EBM Training: Contrastive vs Regularized methods

- **Contrastive methods**
  - Push down on energy of training samples
  - Pull up on energy of suitably-generated contrastive samples
  - Scales very badly with dimension
- **Regularized Methods**
  - Regularizer minimizes the volume of space that can take low energy



Latent-Variable Generative EBM

- **Predicts the desired output y**
- **Handles uncertainty/multi-modality with a latent variable:**
  - parameterizes the set/distribution of plausible predictions.
- **Ideally, the latent variable represents independent explanatory factors of variation**
- **The information capacity of the latent variable must be minimized (with R(z)).**
  - Otherwise all the information for the prediction will go into z → flat energy landscape.

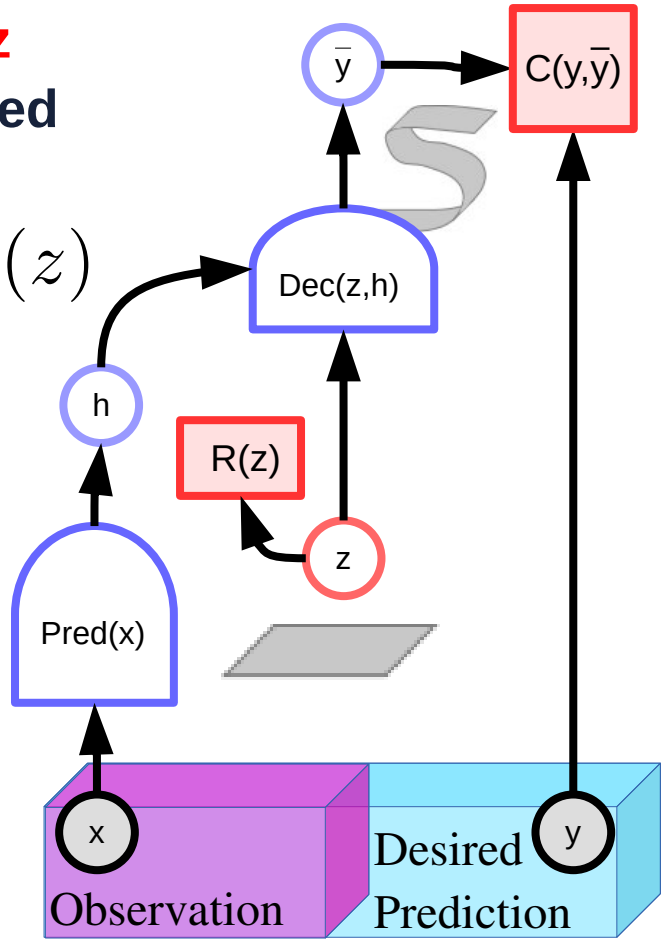


Conditional Latent-Variable EBM

- **Regularizer R(z) limits the information capacity of z**
- **Without regularization, every y may be reconstructed exactly (flat energy surface)**

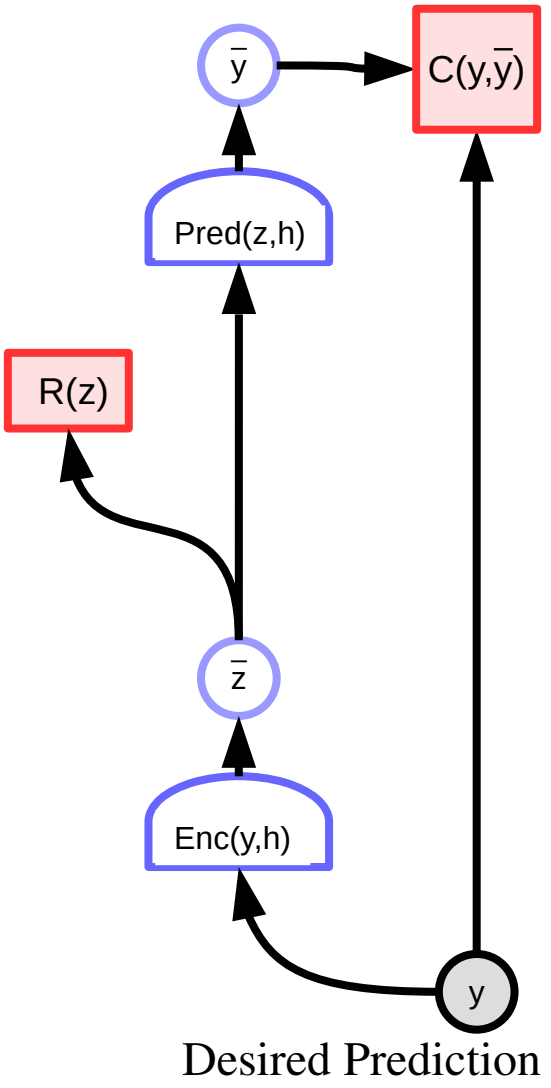
$$E(x, y, z) = C(y, \mathrm{Dec}(\mathrm{Pred}(x), z)) + \lambda R(z)$$

- **Examples of R(z):**
  - Effective dimension [Li et al. NeurIPS 2020]
  - Quantization / discretization
  - L0 norm (# of non-0 components)
  - L1 norm with decoder normalization
  - Maximize lateral inhibition / competition
  - Add noise to z while limiting its L2 norm (VAE)
  - <your_information_throttling_method_goes_here>



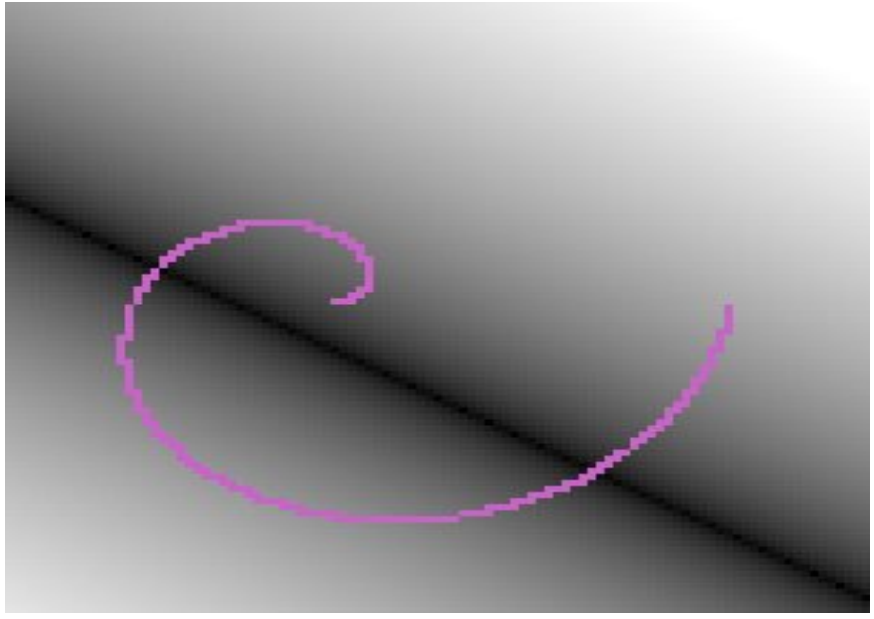
Regularized Auto-Encoders

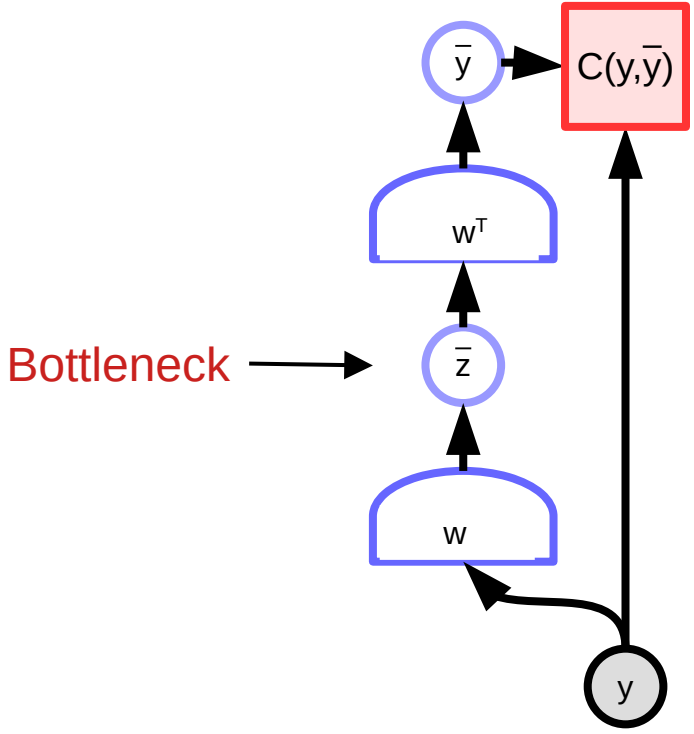
- **Unconditional Models**
- **Regularized Auto-Encoders**
  - Representation computed by encoder
  - No explicit latent variable
- **R(z) now becomes a term in the training loss.**
  - R(z) has no effect on inference
- **Examples:**
  - Bottleneck AE (aka “diabolo” networks)
  - Contracting AE
  - Saturating AE
  - ….



Principal Component Analysis

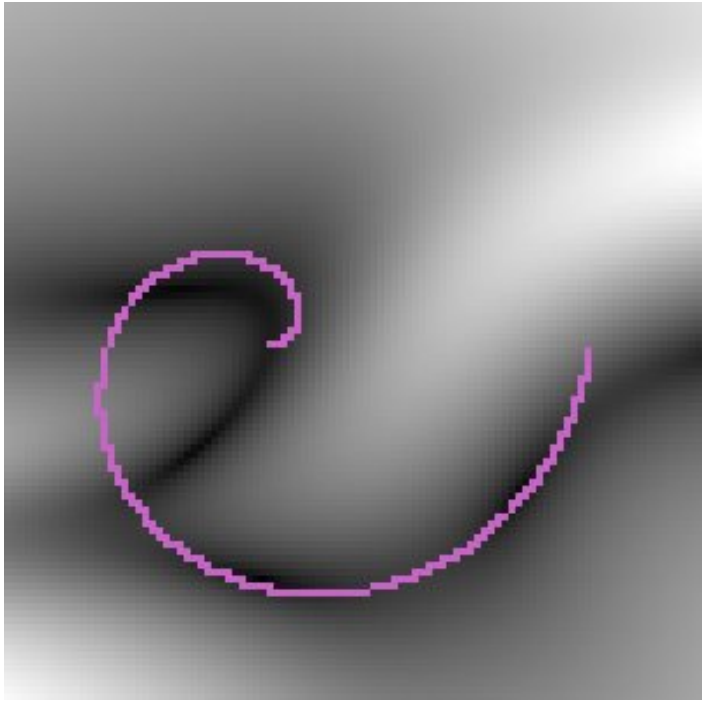
- **PCA is a 2-layer linear auto-encoder with a bottleneck.**
  - Energy: $F_w(y) = ||y - Dec(Enc(y))||^2 = ||y - w^T w||^2$
  - Loss $L(y, w) = F_w(y)$

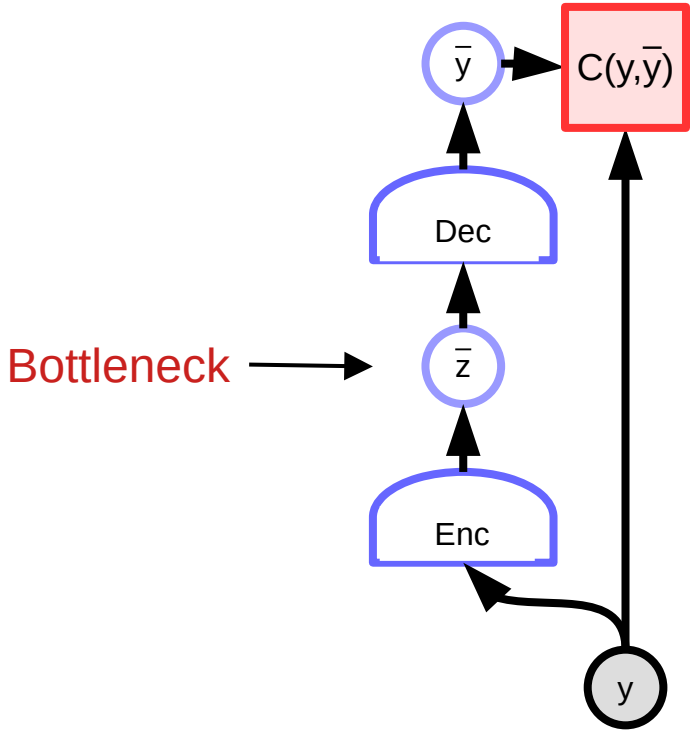




Auto-Encoder with Bottleneck

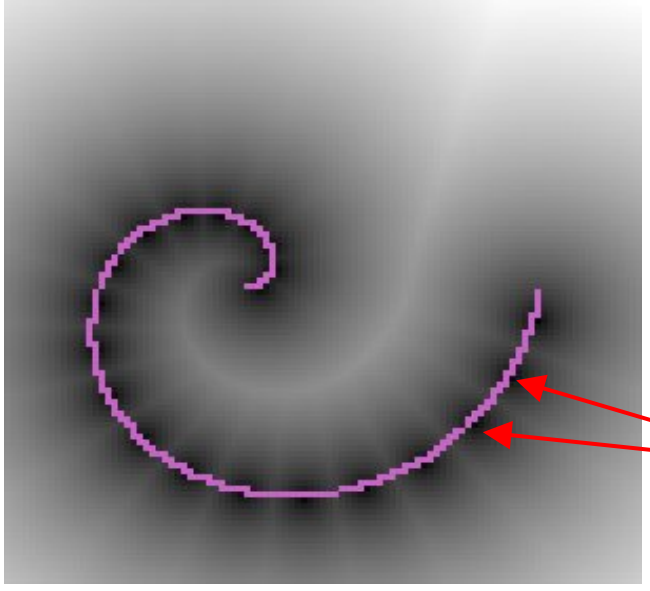
- **non-linear auto-encoder with a bottleneck.**
  - Energy: $F_w(y) = ||y - Dec(Enc(y))||^2$
  - Loss: $L(y, w) = F_w(y)$



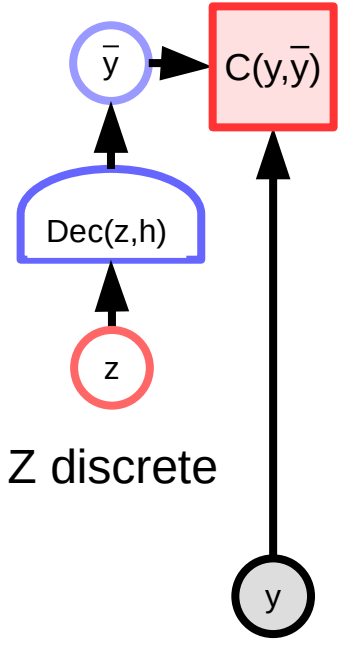


K-Means

- **Discrete latent-variable model with linear decoder**
  - Energy: $E(y,z) = ||y - Dec(z)||^2 = ||y - wz||^2$
  - Free Energy $F(y) = \min_{z \in \mathcal{Z}} E(y,z)$
  - Loss: $L(y,w) = F_w(y)$

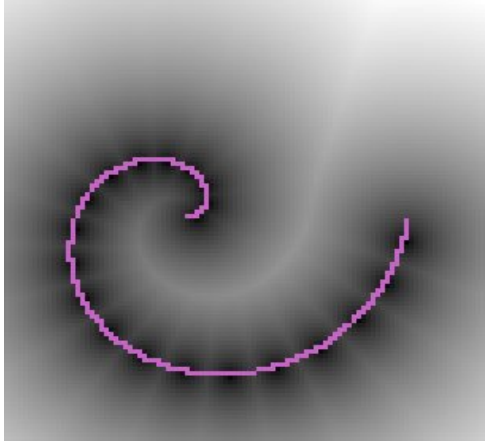


- Latent vector z is constrained to be a 1-hot vector: [0,0,..,01,0,..,0]
- 1 component selects a column of w
- F(y)=0 iff y is equal to a column of w.

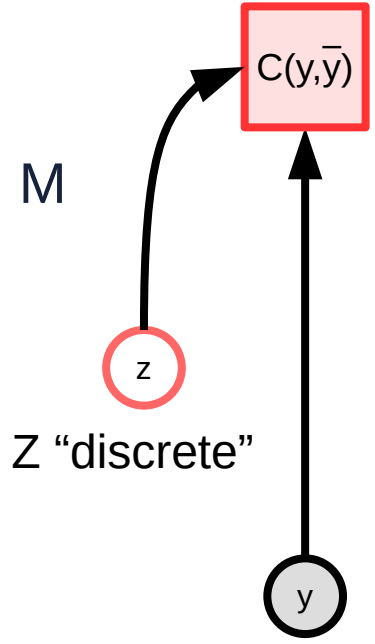


Gaussian Mixture Model

- **Similar to K-means with soft marginalization over latent.**
  - Energy: $E(y,z) = (y - wz)^T (Mz)(y - wz)$ $\quad (Mz)_{ij} = \sum_k M_{ijk} z_k$
  - Free Energy $\quad F(y) = -\frac{1}{\beta} \log \sum_{z \in \mathcal{Z}} e^{\beta E(y,z)}$
  - Loss: $L(y,w) = F_w(y)$ with normalization constraint on M



- Latent vector z is constrained to be a 1-hot vector: [0,0,..,01,0,..,0]
- But marginalization makes it “soft”

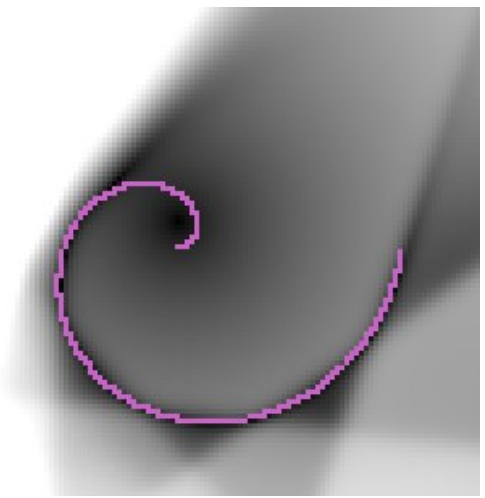


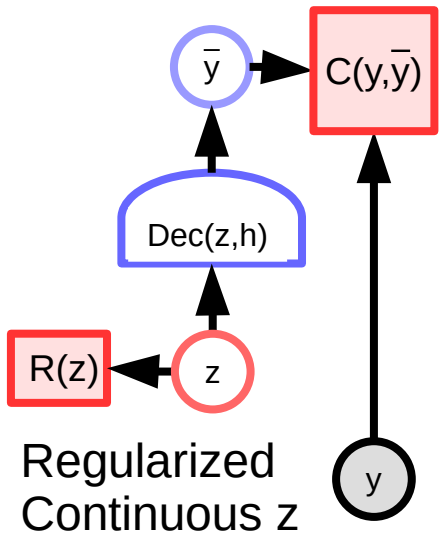
Regularized Latent Variable: Sparse Coding

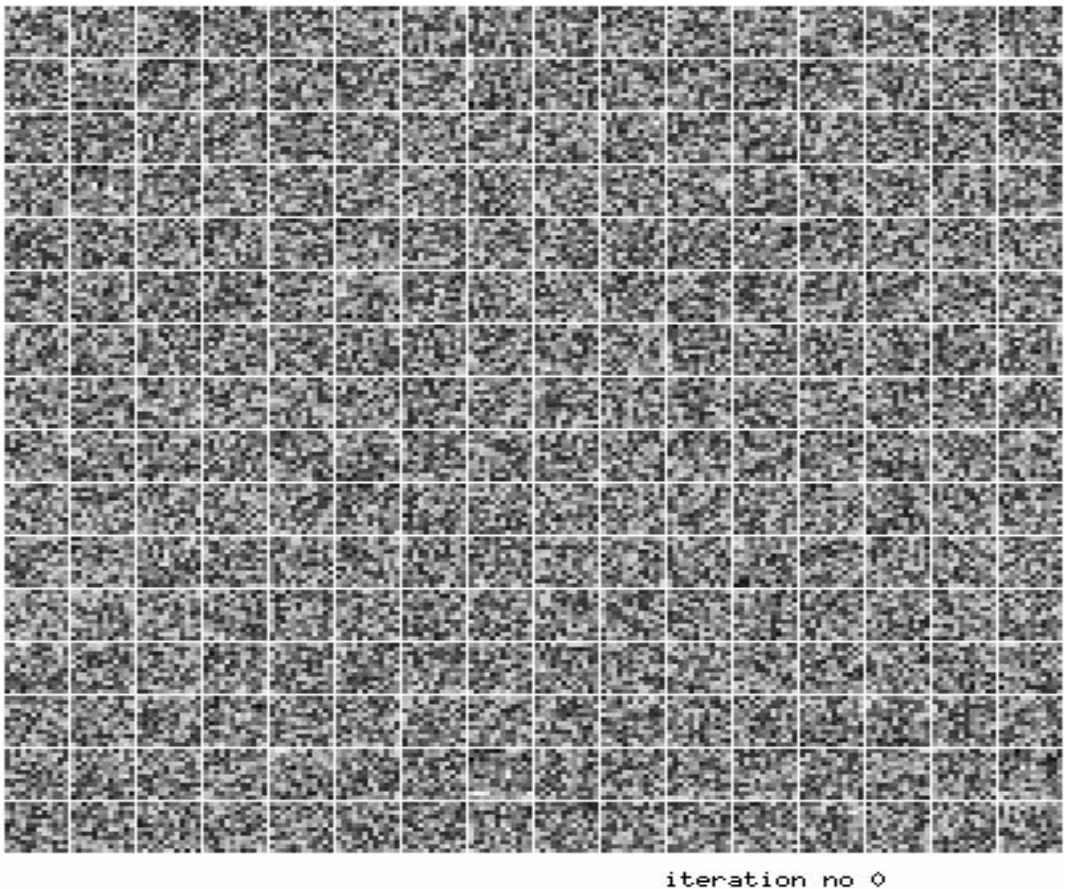
- **A2:** **regularize the volume of the low energy regions**

$$E(y,z) = ||y - wz||^2 + \lambda |z|_{L1|}$$

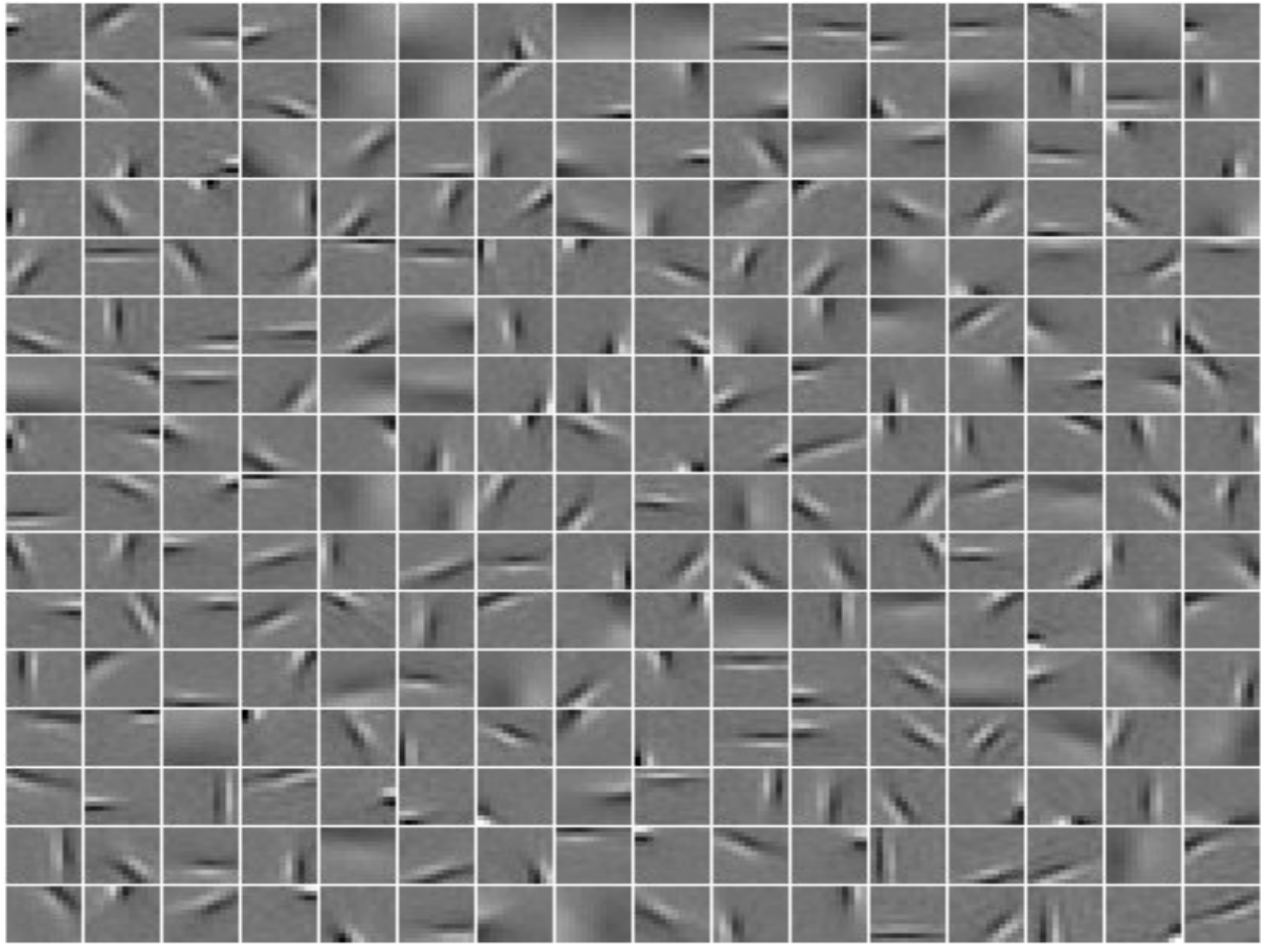
$$F(y) = \min_z E(y,z)$$





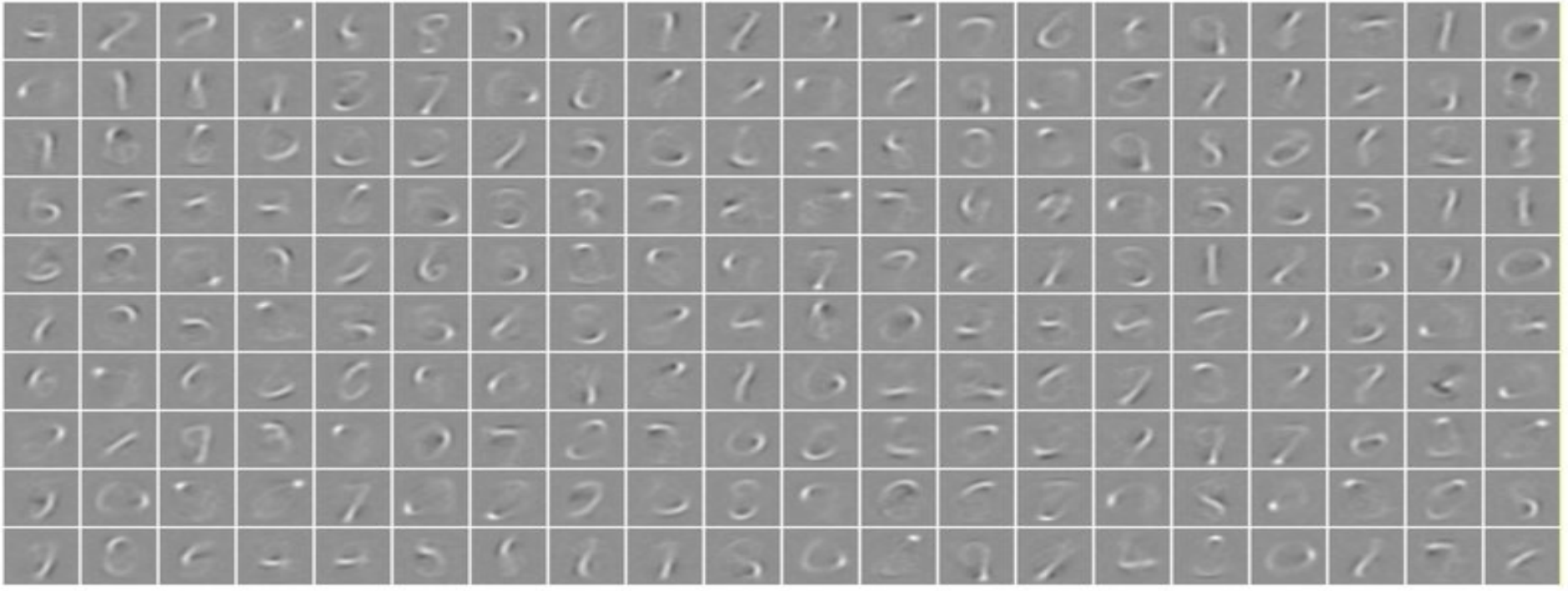


Learned Features on natural patches: V1-like receptive fields



Sparse Modeling on handwritten digits (MNIST)

- **Basis functions (columns of decoder matrix) are digit parts**
- **All digits are a linear combination of a small number of these**



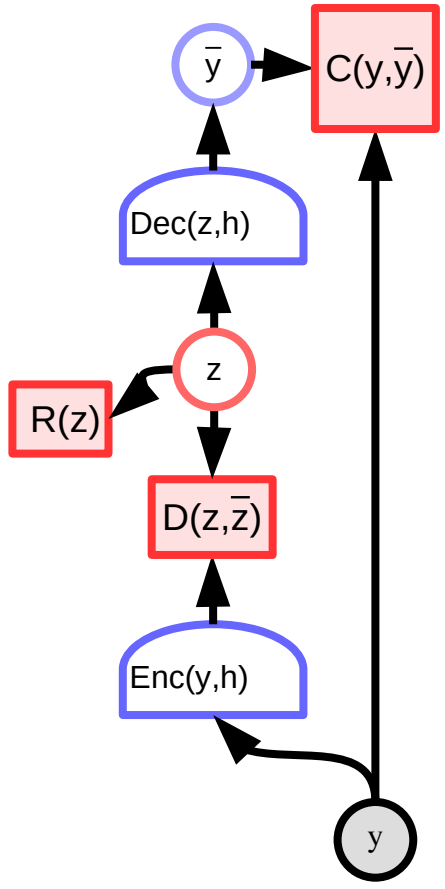
Amortized Inference

- **Training an encoder to give an approximate solution to the inference optimization problem**
- **Regularized Auto-Encoder, Sparse AE, LISTA**

$$\mathrm{E(y,z)} = \mathrm{C(y,\ Dec((z))} + \mathrm{D(z,Enc(y))} + \lambda R(z)$$

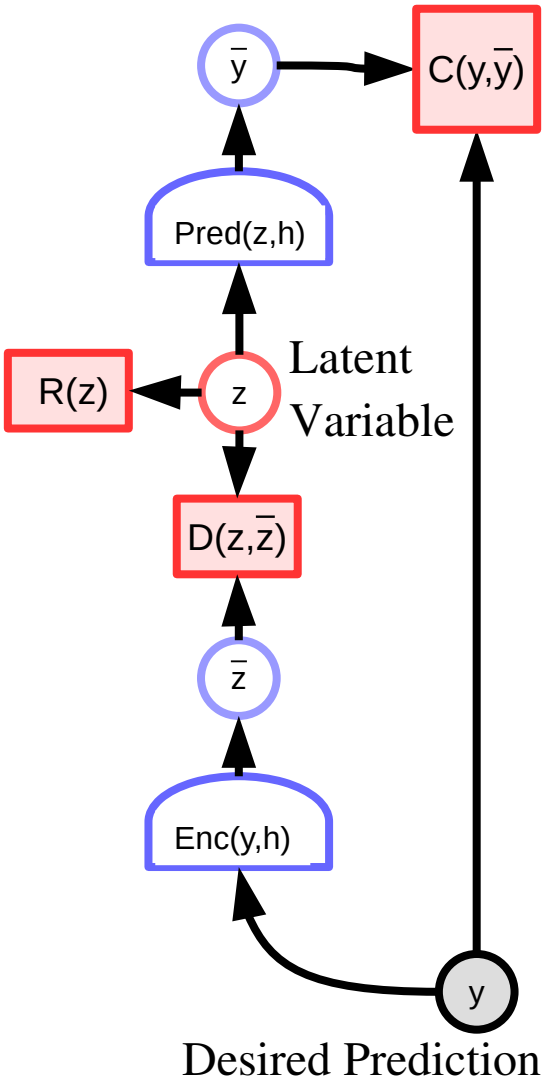
$$F(y) = \min_z E(y, z)$$

- **"A tutorial on amortized optimization…" by Brendan Amos et al. arxiv:2202.00665**



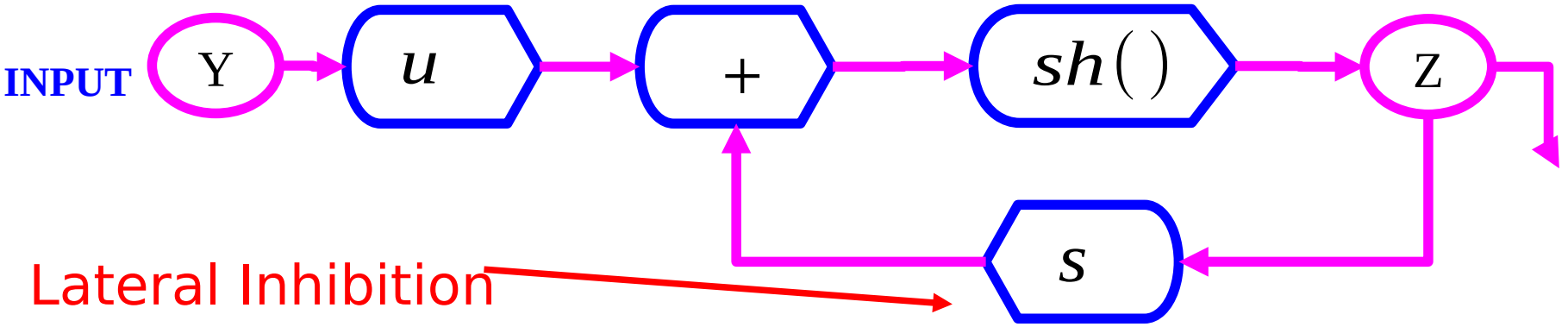
Unconditional LVGEBM with Amortized Inference

- **Latent-Variable Regularized Auto-Encoders**
  - Variational AE
  - Predictive Sparse Coding
  - ….
- **Energy has three terms:**
  - Reconstruction $C(y,\bar{y})$
  - latent regularization $R(z)$
  - latent prediction $D(z,\bar{z})$
- **Inference:**
  - Initialize z to $\bar{z}$
  - Find z that minimizes $E(y,z)=C(y,\bar{y})+R(z)+D(z,\bar{z})$



Giving the “right” structure to the encoder

- **ISTA/FISTA: iterative algorithm that converges to optimal sparse code**



$$z(t+1) = \text{Shrink}_{\alpha\eta}\left[z(t) - \eta w^t(wz(t) - y)\right]$$

- **ISTA/FastISTA reparameterized:**

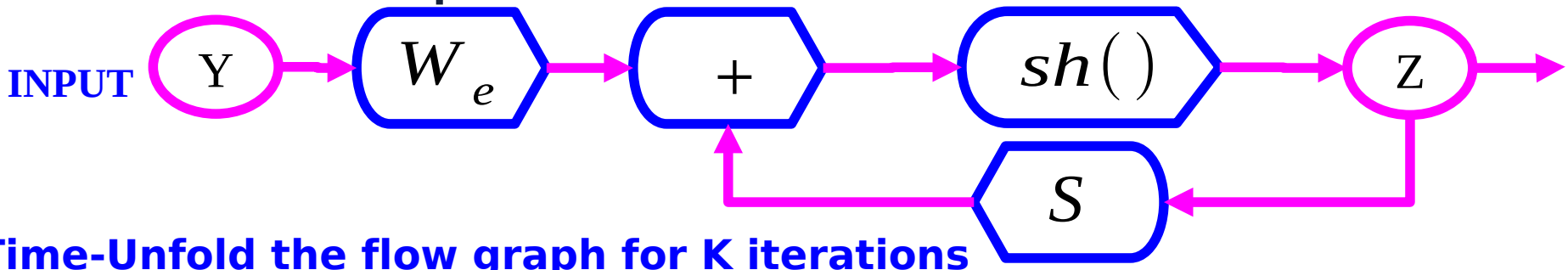
$$z(t+1) = \text{Shrink}_{\alpha\eta}\left[sz(t) + uy\right]; \quad u = \eta w; \quad s = I - \eta w^T w$$

- **LISTA (Learned ISTA): learn the We and S matrices to get fast solutions**

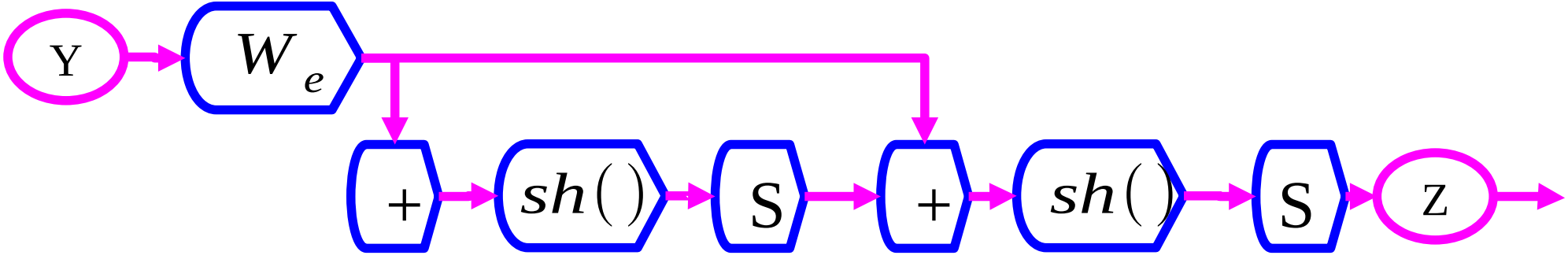
[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

LISTA: Train We and S matrices to give a good approximation quickly

- **Think of the FISTA flow graph as a recurrent neural net where We and S are trainable parameters**

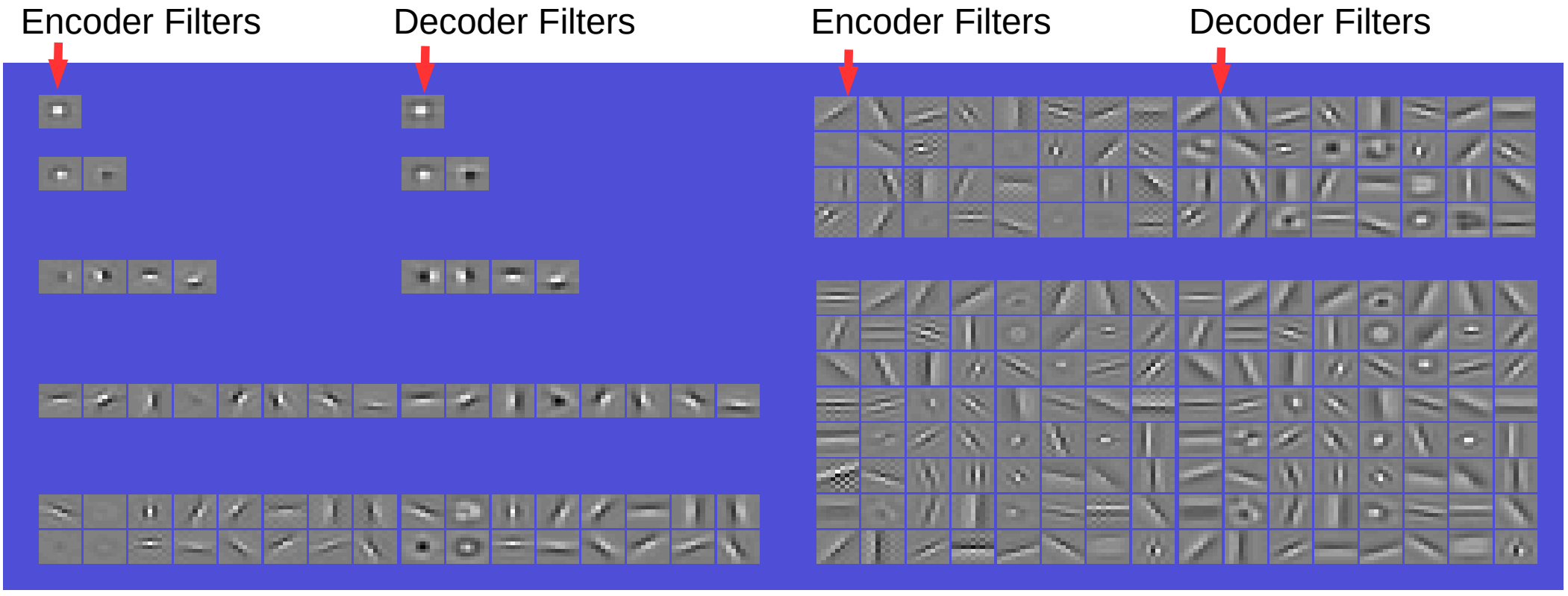


- **Time-Unfold the flow graph for K iterations**
- **Learn the We and S matrices with "backprop-through-time"**
- **Get the best approximate solution within K iterations**



Convolutional Sparse Auto-Encoder on Natural Images

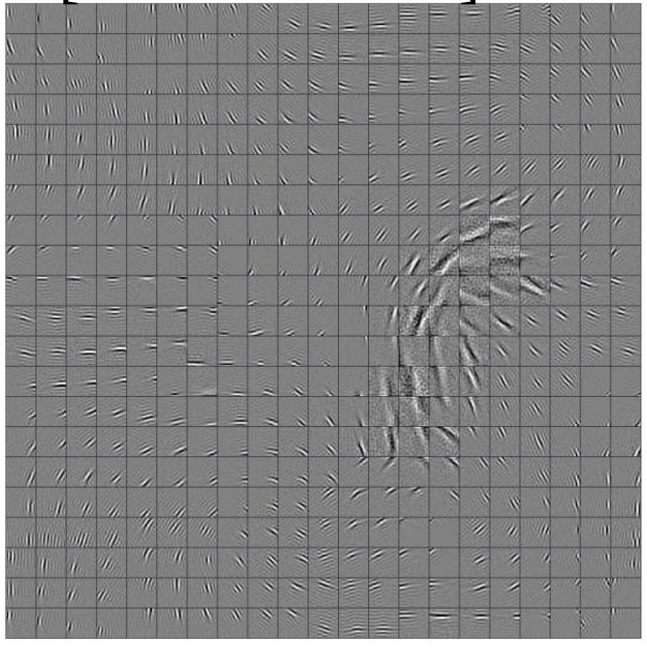
- **Filters and Basis Functions obtained. Linear decoder (conv)**
  - with 1, 2, 4, 8, 16, 32, and 64 filters [Kavukcuoglu NIPS 2010]



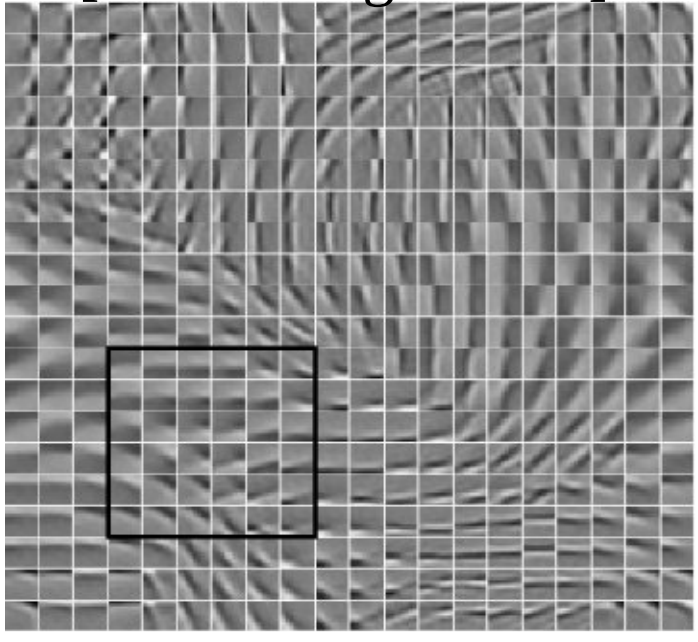
Learning invariant features

- **Sparsity over pooling units → group sparsity**
  - [Hyvarinen & Hoyer 2001], [Osindero et al. Neural Comp. 2006], [Kavukcuoglu et al. CVPR 2009], [Gregor & LeCun arXiv:1006.044], [Mairal et al. 2011].

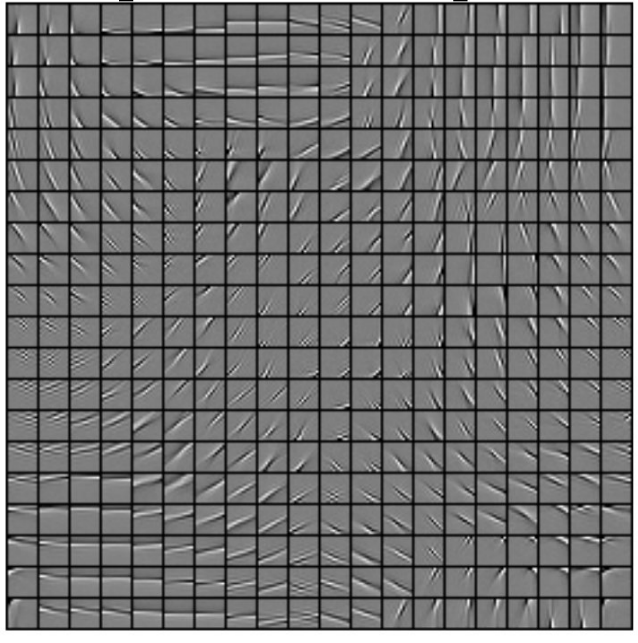
[Osindero 2006]



[Kavukcuoglu 2009]

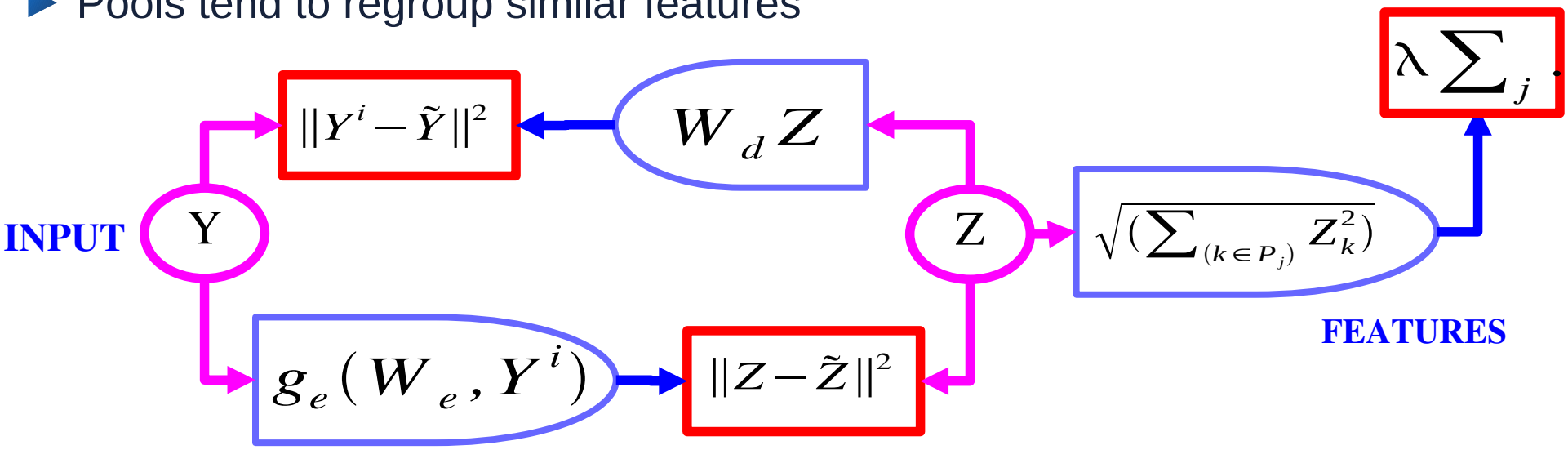


[Mairal 2011]



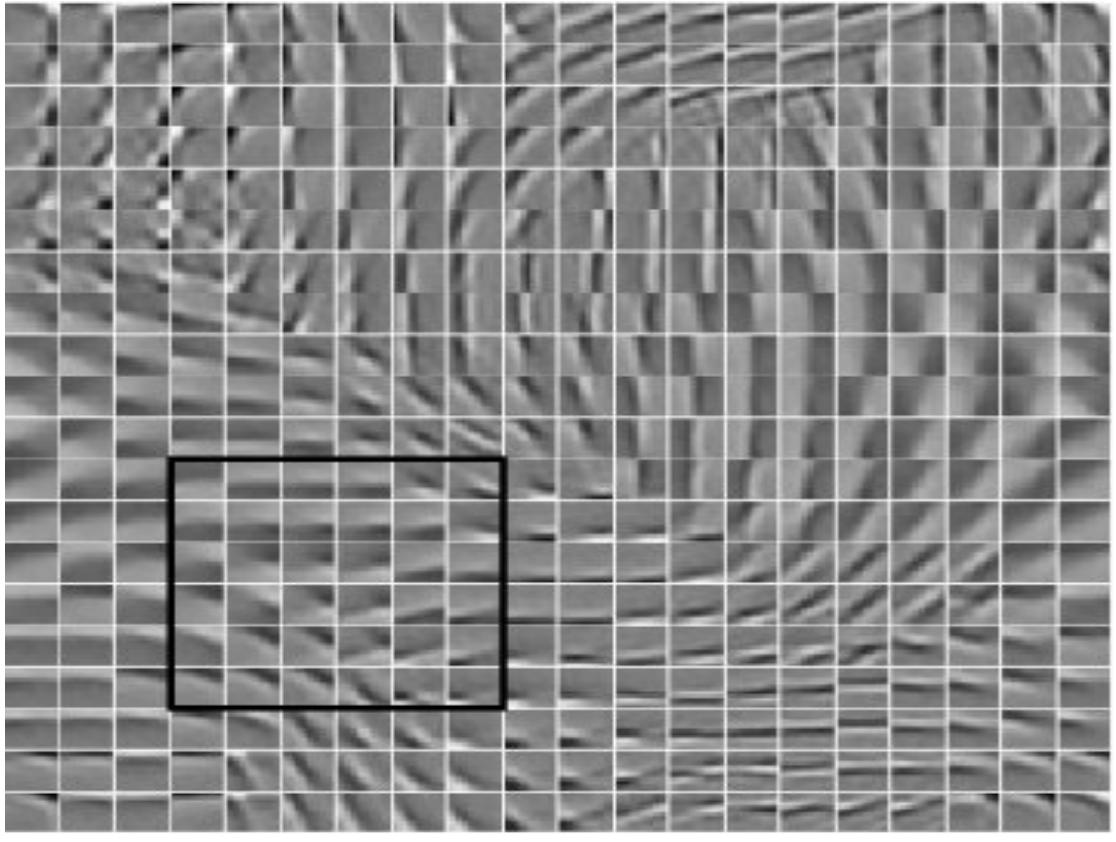
AE with Group Sparsity [Kavukcuoglu et al. CVPR 2009]

- **Trains a sparse AE to produce invariant features**
- **Could we devise a similar method that learns the pooling layer as well?**
- **Group sparsity on pools of features**
  - Minimum number of pools must be non-zero
  - Number of features that are on within a pool doesn't matter
  - Pools tend to regroup similar features



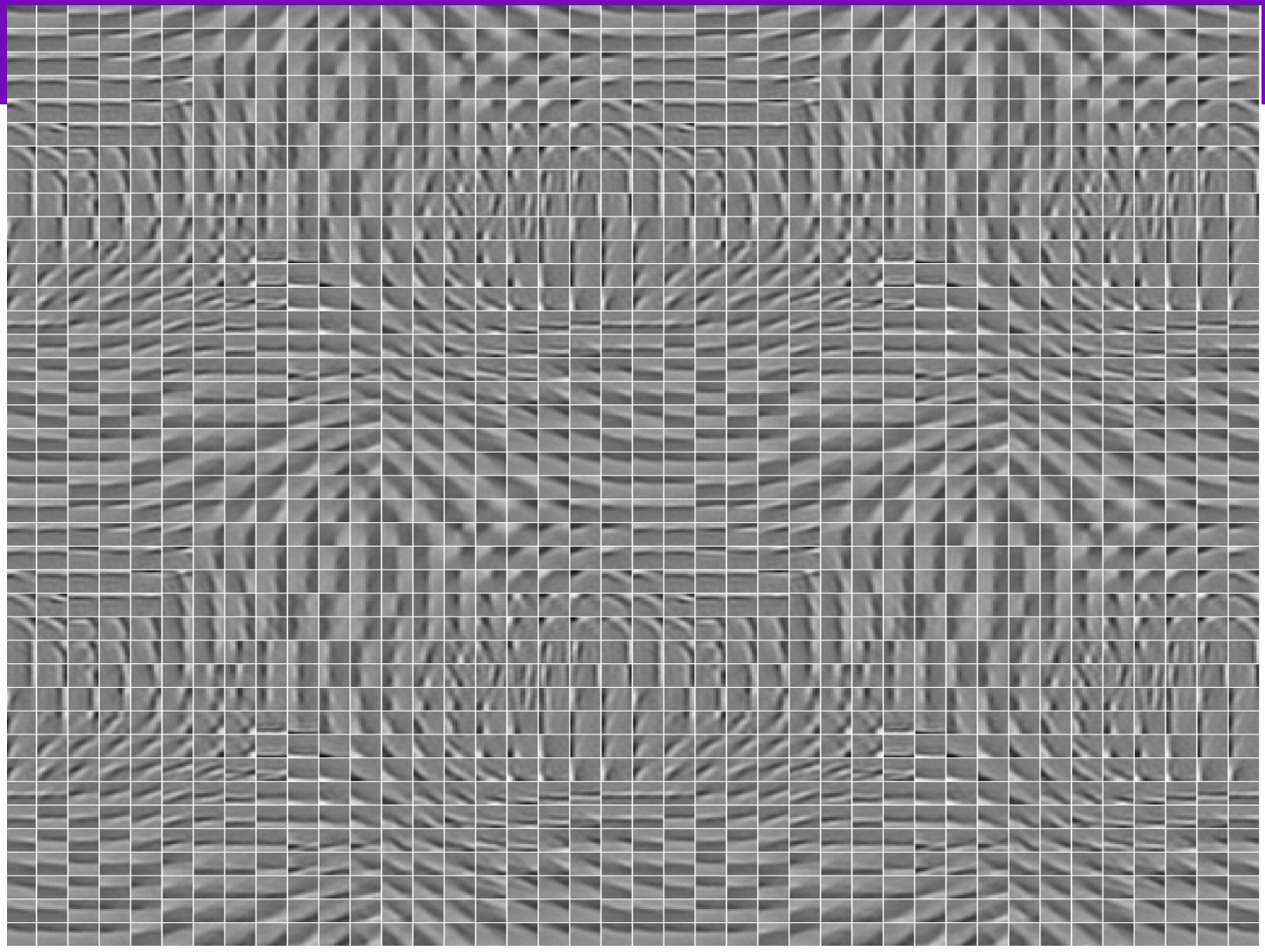
Pooling over features in a topographic map.

- **The pools are 6x6 blocks of features arranged in a 2D torus**
- **While training, the filters arrange themselves spontaneously so that similar filters enter the same pool.**
- **The pooling units can be seen as complex cells**
- **They are invariant to local transformations of the input**
  - For some it's translations, for others rotations, or other transformations.



Pinwheels!

- **The so-called “pinwheel” pattern is observed in the primary visual cortex.**



- **End of lecture 009**