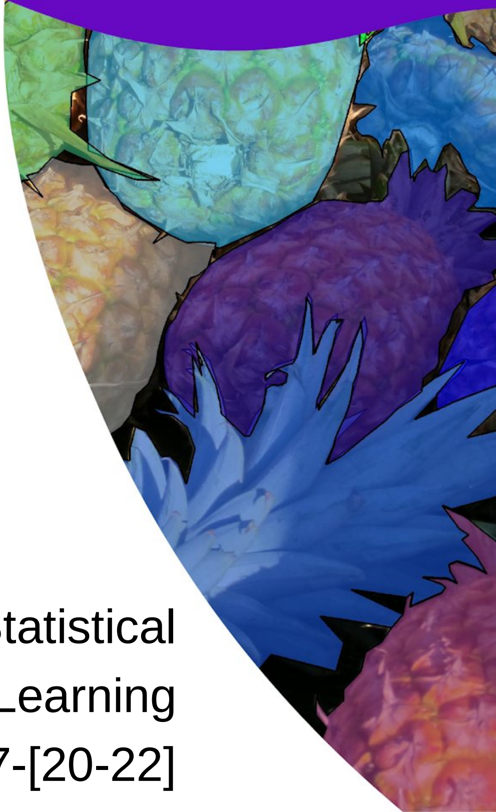
NEW YORK UNIVERSITY

From Machine Learning to Autonomous Intelligence Lecture 3

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



Summer School on Statistical Physics & Machine Learning
Les Houches, 2022-07-[20-22]

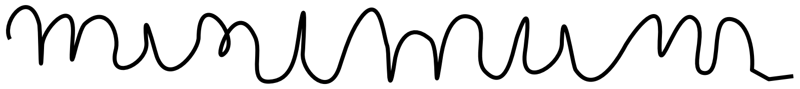
FACEBOOK AI

Latent-Variable Models for (supervised) structured prediction.

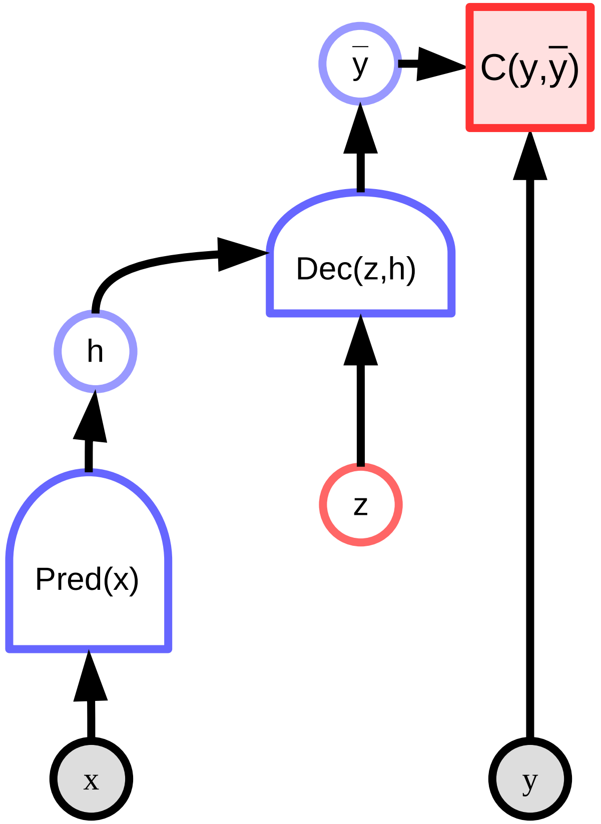
Structured prediction with latent variable models

When inference involves latent variables

- **Latent variables are variables whose value is never given to us.**
  - Examples: to read a handwritten word, it helps to know where the characters are

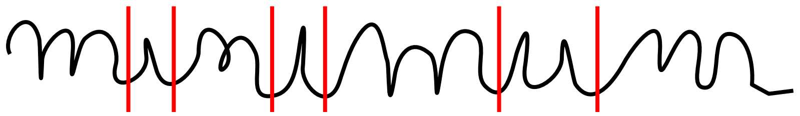


  - To recognize speech, it helps to know where the words and phonemes are
    - Youcantreadthisifyoudontunderstandenglish
    - Vousnepouvezpaslirececisivousneparlezpasfrançais

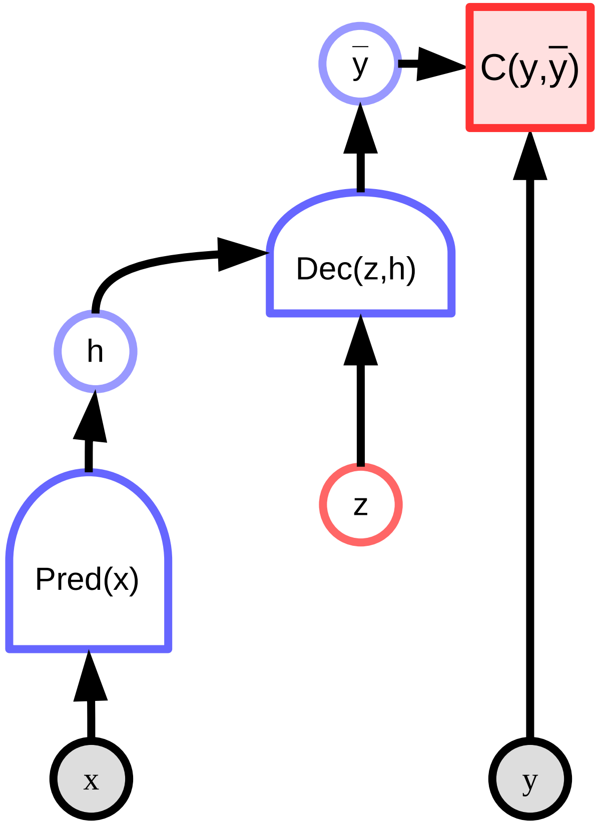


When inference involves latent variables

- **Latent variables are variables whose value is never given to us.**
  - Examples: to read a handwritten word, it helps to know where the characters are

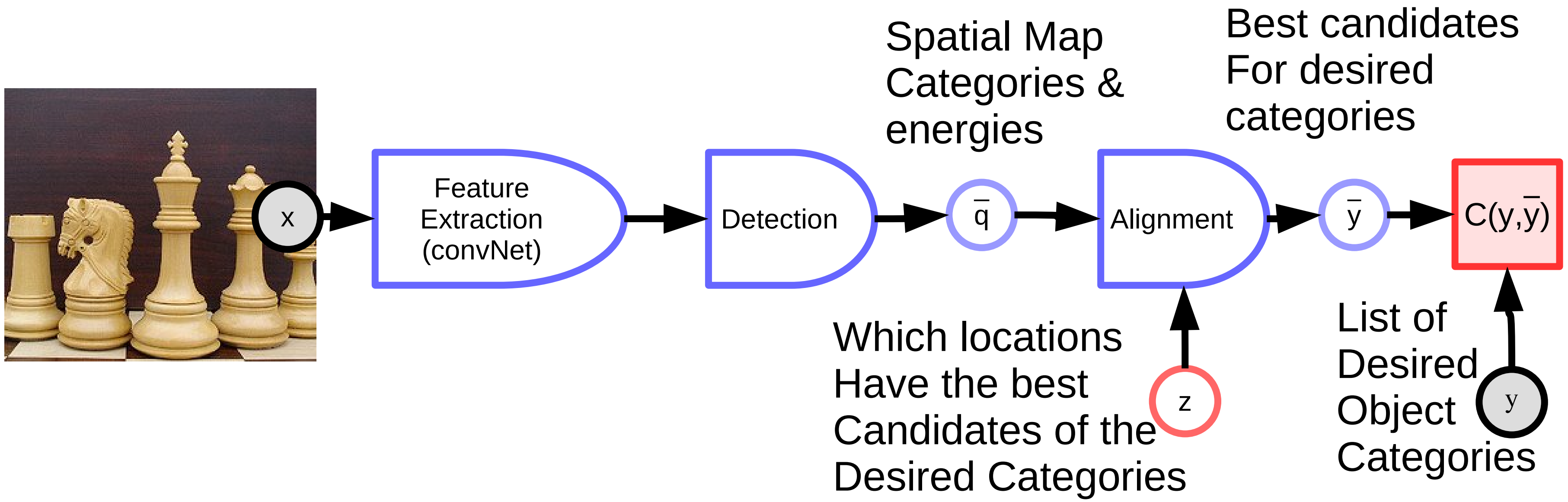


  - To recognize speech, it helps to know where the words and phonemes are
    - You can’t read this if you don’t understand english
    - Vous ne pouvez pas lire ceci si vous ne parlez pas français



Structured Prediction

- **Complex output with weak/partial supervision**
  - Speech recognition: alignment of audio to text transcription
  - Handwriting recognition: alignment of image to character sequence
  - Object detection: alignment of image features to labeled categories


x
Feature Extraction (convNet)
Detection
Spatial Map Categories & energies
$\bar{q}$
Alignment
Best candidates For desired categories
$\bar{y}$
$C(y,\bar{y})$
Which locations Have the best Candidates of the Desired Categories
z
List of Desired Object Categories
y

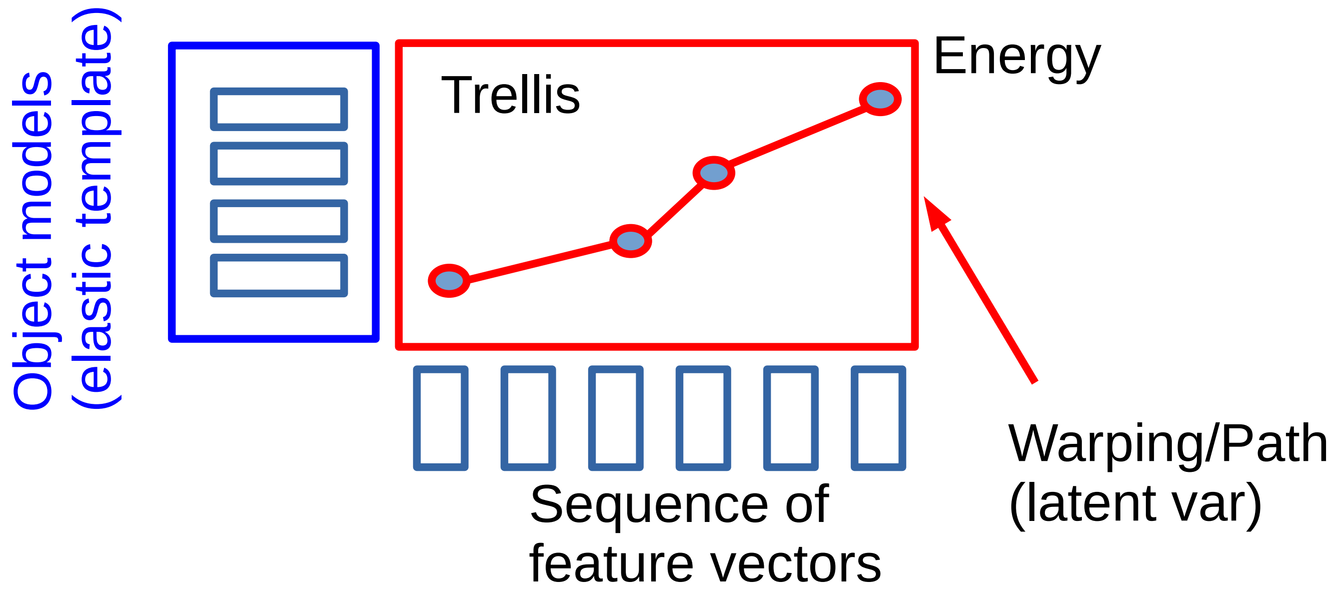
Example: Elastic matching (in 1D = Dynamic Time Warping)

Spoken word recognition with trainable elastic templates and trainable feature extraction

[Driancourt&Bottou 1991, Bottou 1991, Driancourt 1994]

Elastic matching using dynamic time warping (Viterbi algorithm on a trellis).

The corresponding EBFG is implicit (it changes for every new sample).

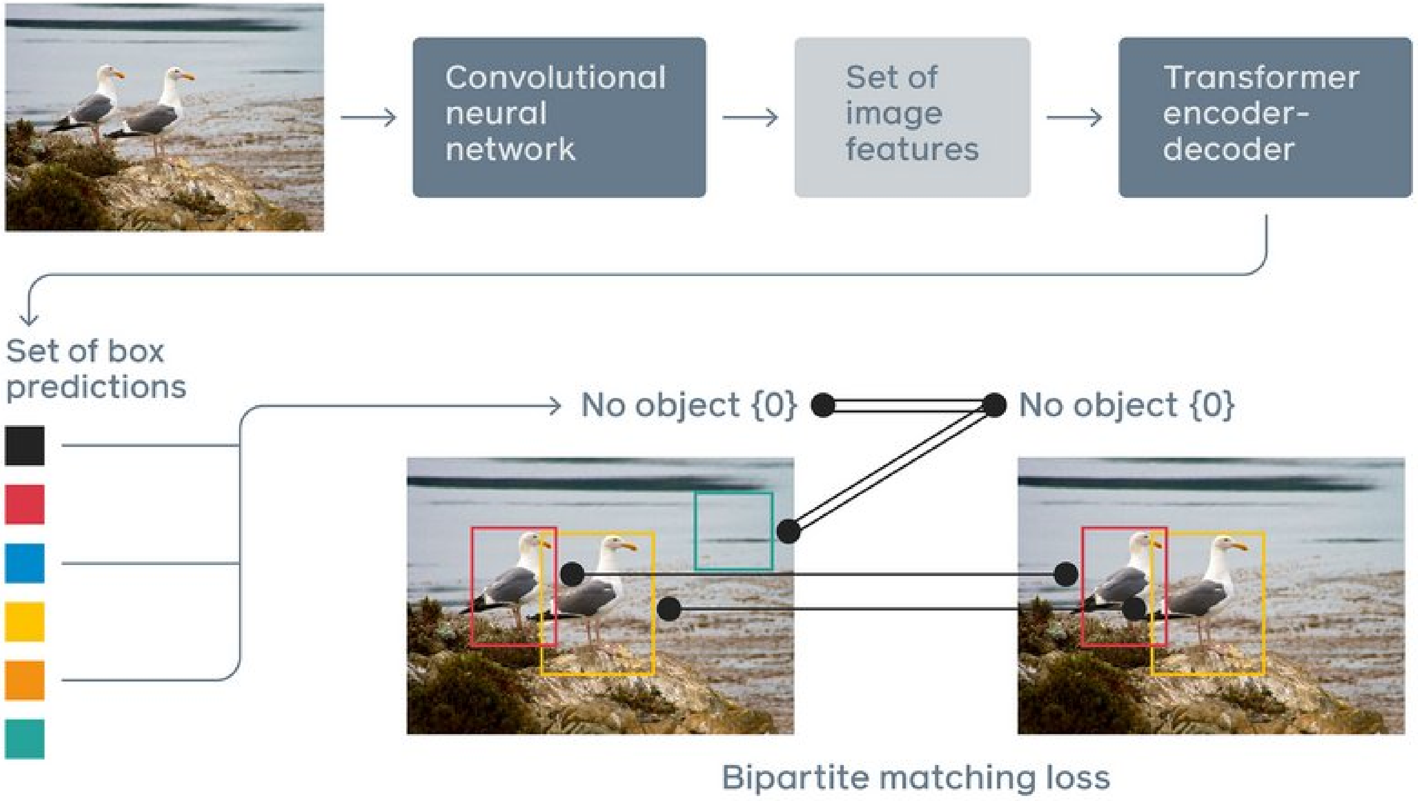


What can the latent variables represent?

- **Variables that would make the task easier if they were known:**
  - **Face recognition**: the gender of the person, the orientation of the face.
  - **Object recognition**: the pose parameters of the object (location, orientation, scale), the lighting conditions.
  - **Parts of Speech Tagging**: the segmentation of the sentence into syntactic units, the parse tree.
  - **Speech Recognition**: the segmentation of the sentence into phonemes or phones.
  - **Handwriting Recognition**: the segmentation of the line into characters.
  - **Object Recognition/Scene Parsing:** the segmentation of the image into components (objects, parts,…), assignment of labels to objects.
- **In general, we will search for the value of the latent variable that allows us to get an answer (y) of smallest energy.**

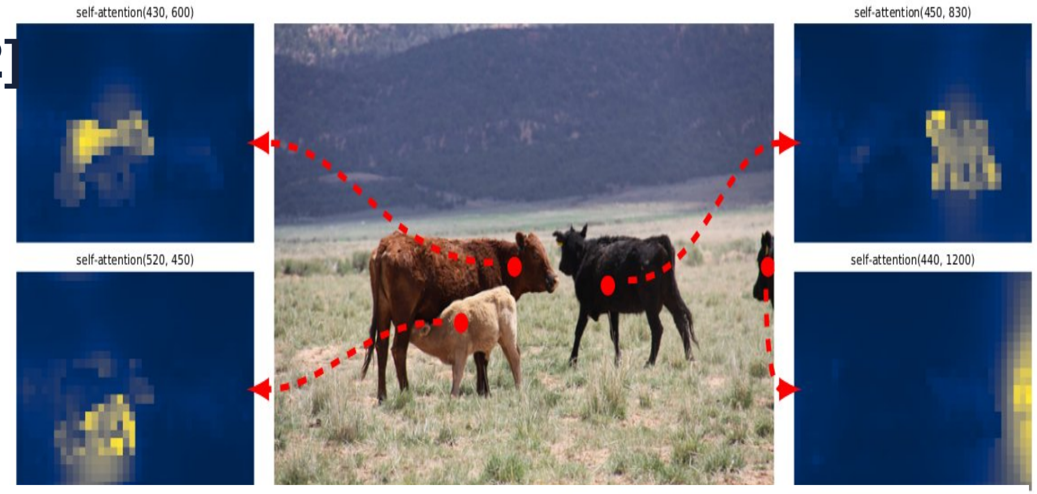
DETR:

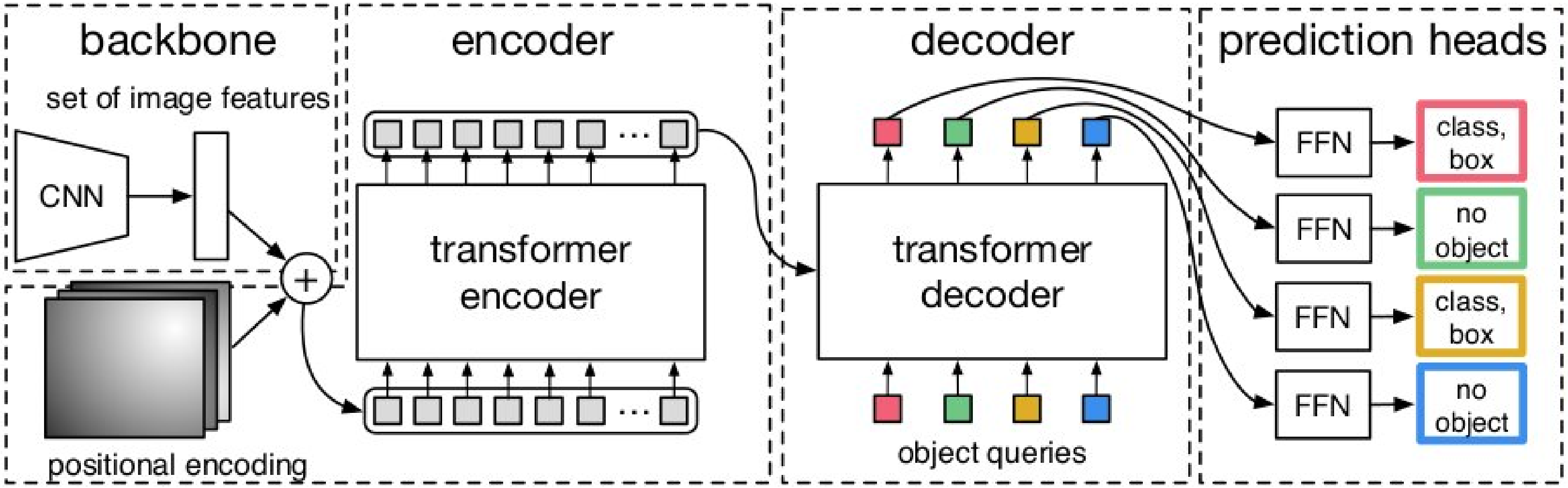
- **DETR [Carion et al. ArXiv:2005.12872] https://github.com/facebookresearch/detr**
- **ConvNet → Transformer**
- **Object-based visual reasoning**

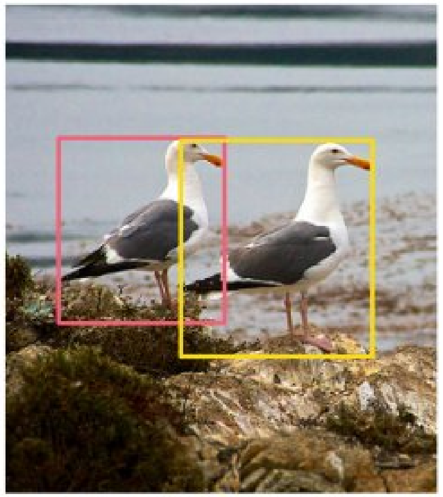


DETR: ConvNet → Transformer for object detection

- **DETR [Carion et al. ArXiv:2005.12872]**
- **https://github.com/facebookresearch/detr**
- **ConvNet → Transformer**
- **Object-based visual reasoning**
- **Transformer: dynamic networks**
  - Through "attention"

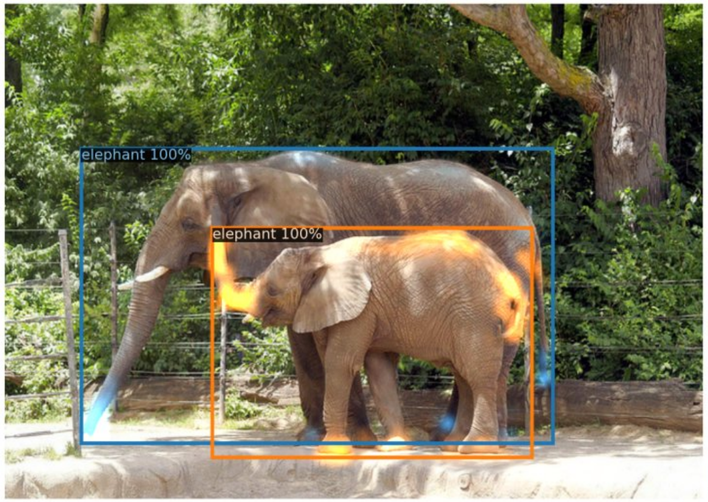


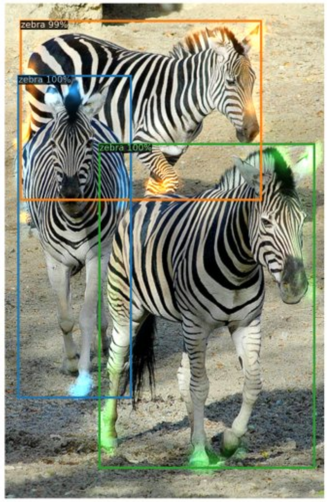


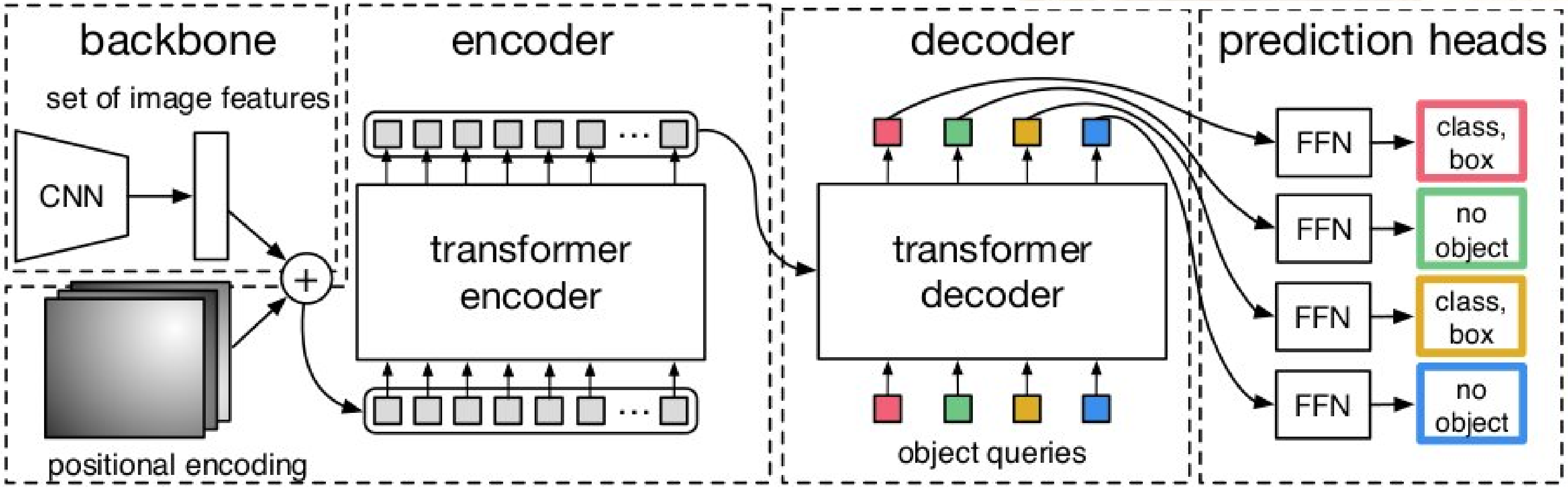


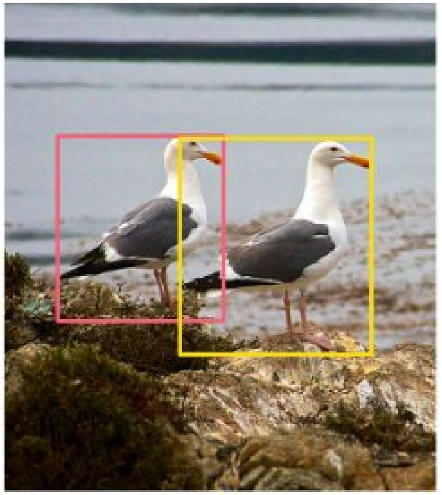
DETR: ConvNet → Transformer for object detection

- **DETR [Carion et al. ArXiv:2005.12872**
- **https://github.com/facebookresearch/detr**
- **ConvNet → Transformer**
- **Object-based visual reasoning**
- **Transformer: dynamic networks**
  - Through "attention"

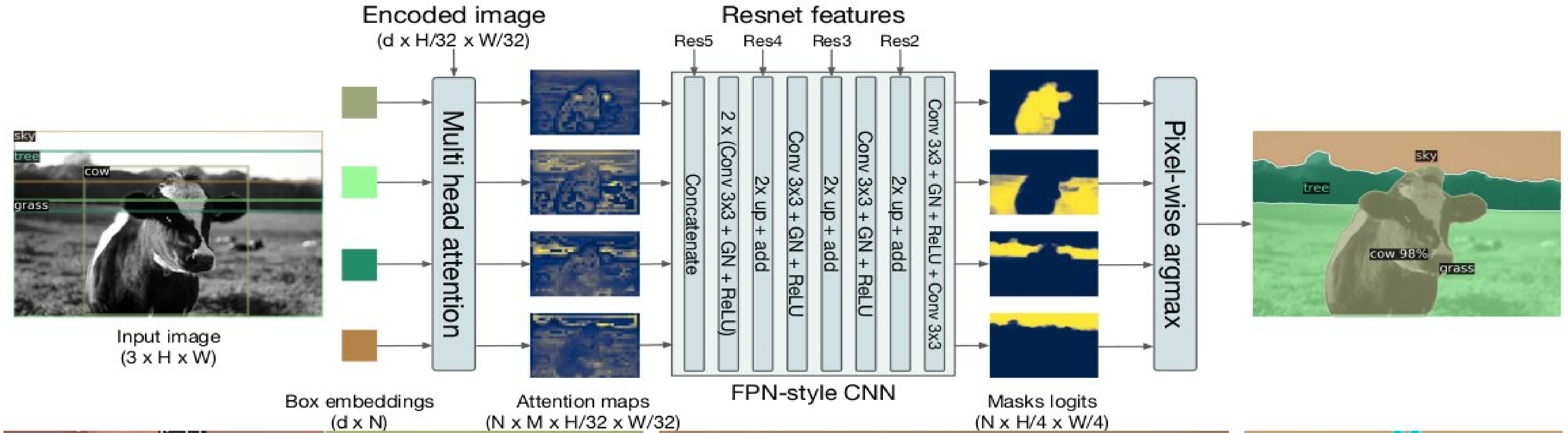


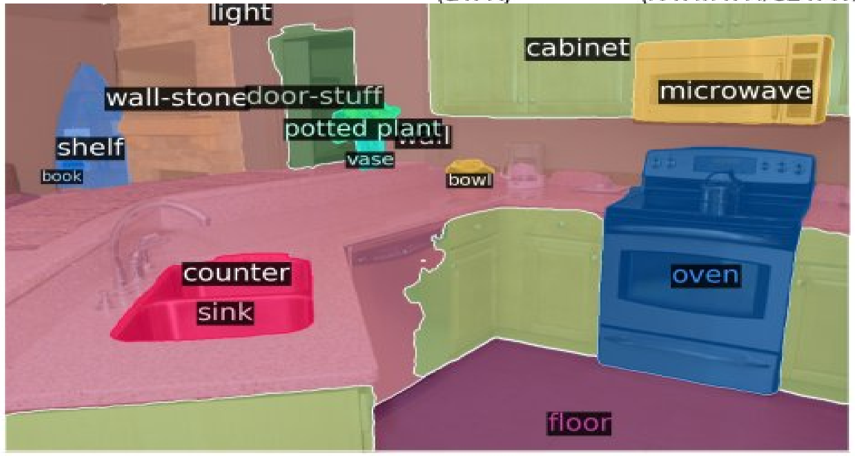


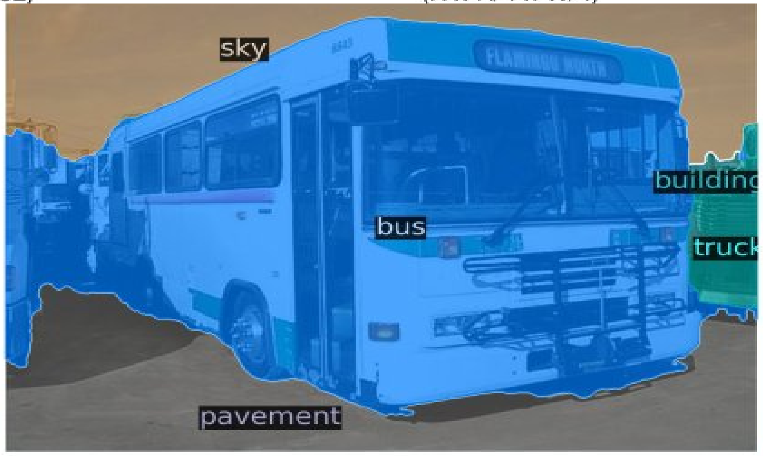


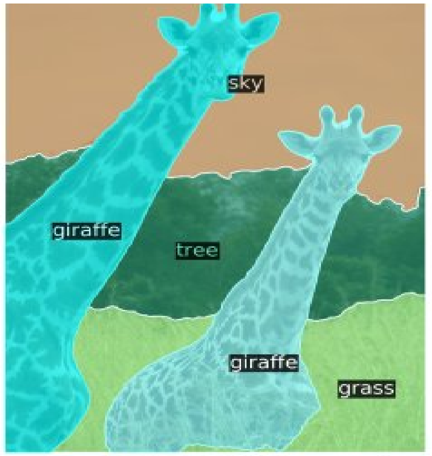


DETR: results on panoptic segmentation





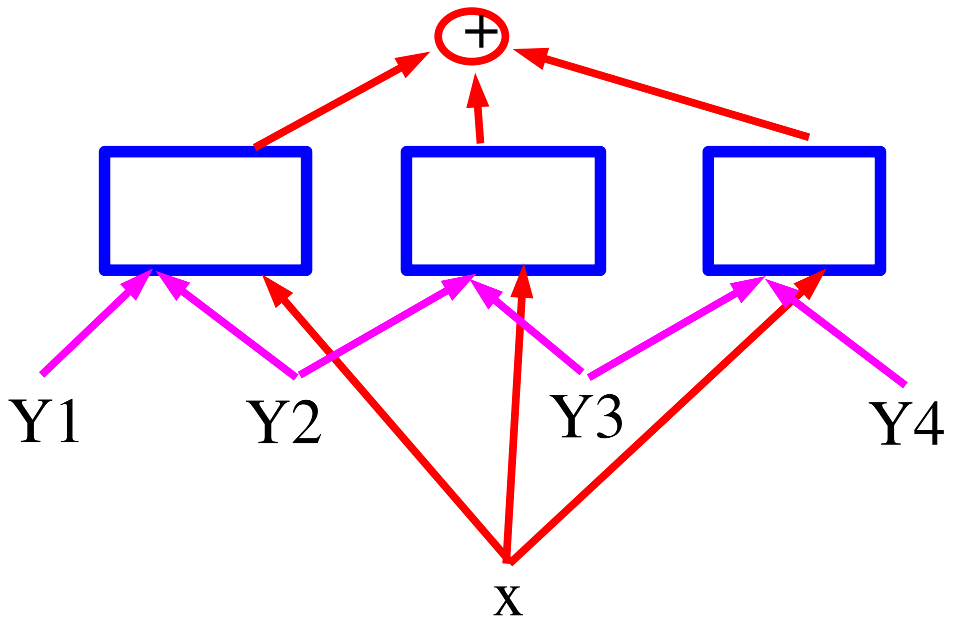




Energy-Based Factor Graphs: Energy = Sum of “factors”

- **Sequence Labeling**
  - Output is a sequence Y1,Y2,Y3,Y4......
  - NLP parsing, MT, speech/handwriting recognition, biological sequence analysis
  - The factors ensure grammatical consistency
  - They give low energy to consistent sub-sequences of output symbols
  - The graph is generally simple (chain or tree)
  - Inference is easy (dynamic programming, min-sum)

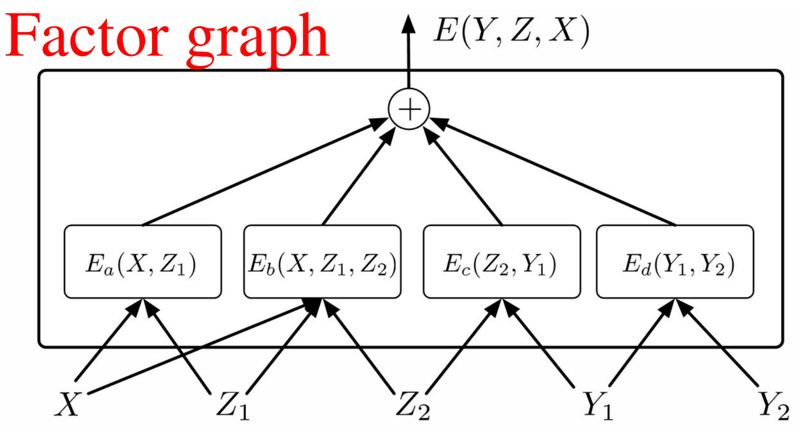
$$\check{y} = \text{argmin}_{z \in \mathcal{Z}, y \in \mathcal{Y}} E(x, y, z)$$

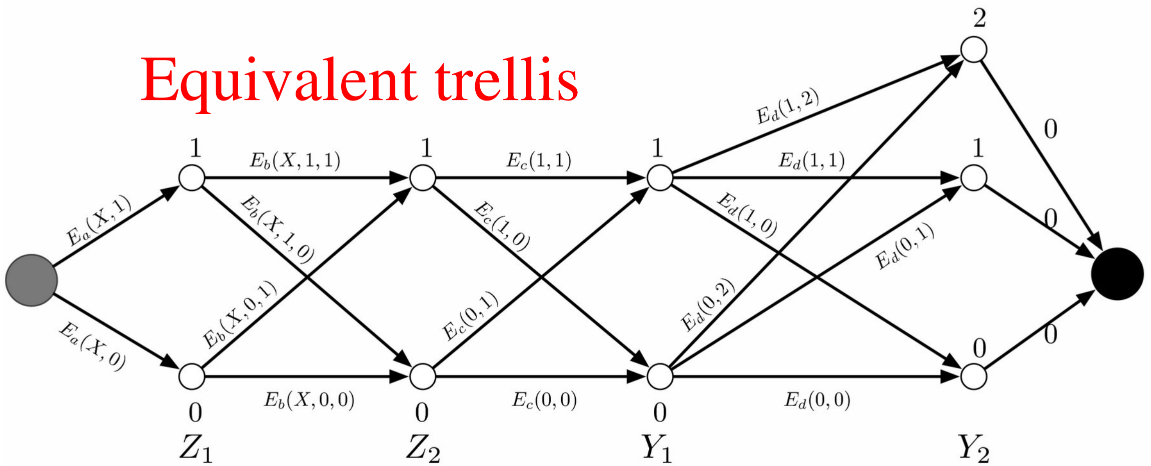


Efficient Inference: Energy-Based Factor Graphs

- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3=24 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 6.
  - Hence, we can precompute the 16 factor values, and put them on the arcs in a trellis.
  - A path in the trellis is a config of variable
  - The cost of the path is the energy of the config

- **The energy is a sum of “factor” functions**





NEW YORK UNIVERSITY

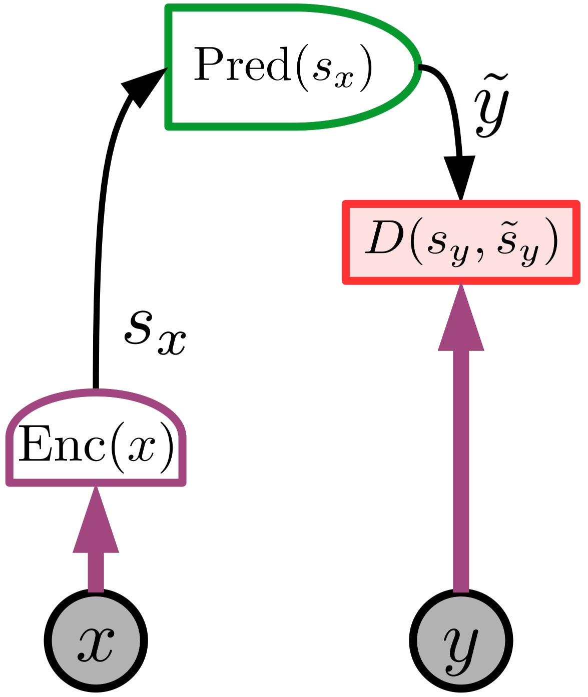
EBM Architectures

for multimodal prediction

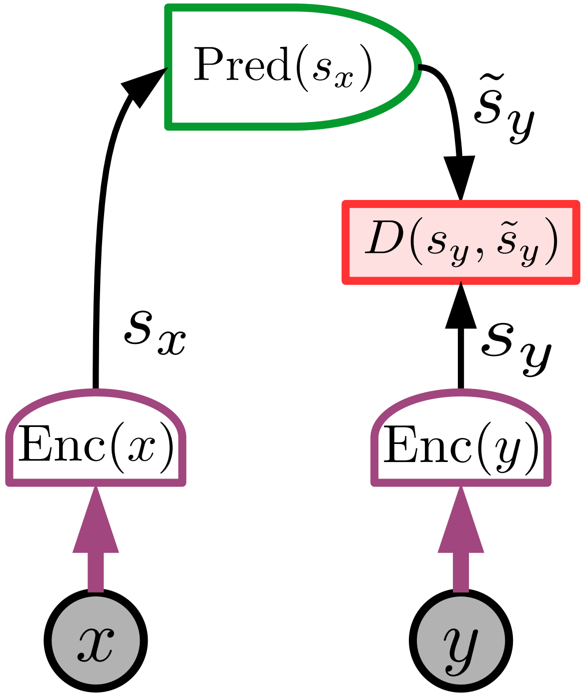
Latent variable models
Joint embedding architectures
Joint Embedding Predictive Architecture (JEPA)

EBM Architectures: Generative vs Joint Embedding

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



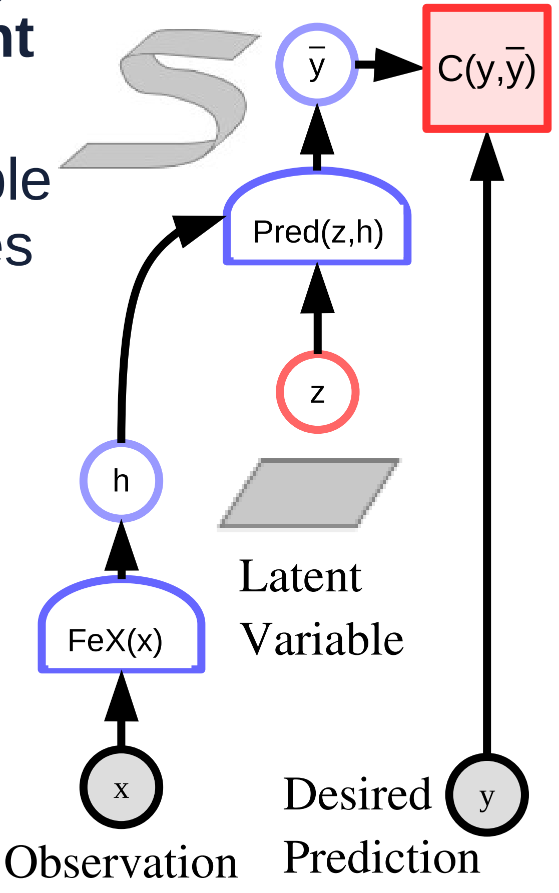
a) Generative Architecture

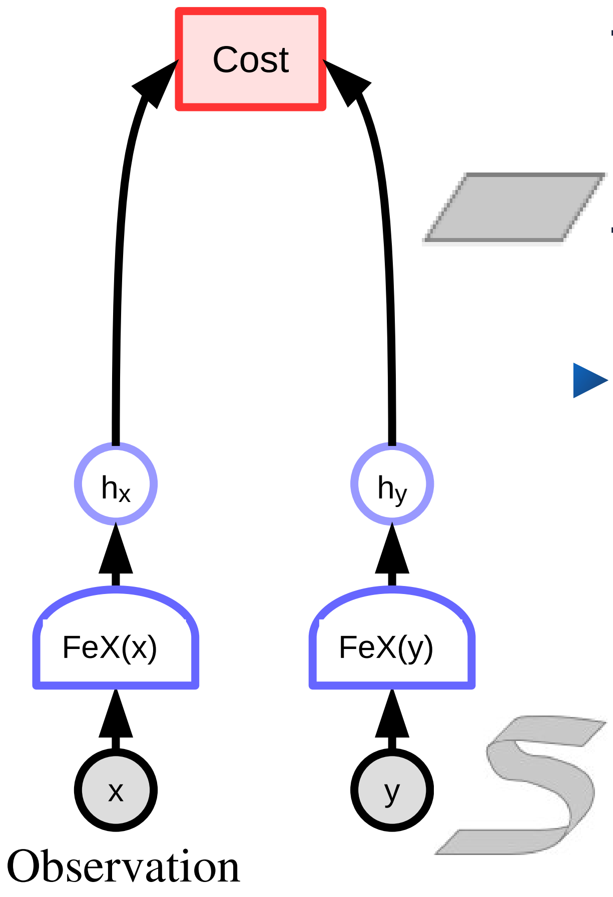


b) Joint Embedding Architecture

Latent-Variable Generative EBM / Joint Embedding EBM

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

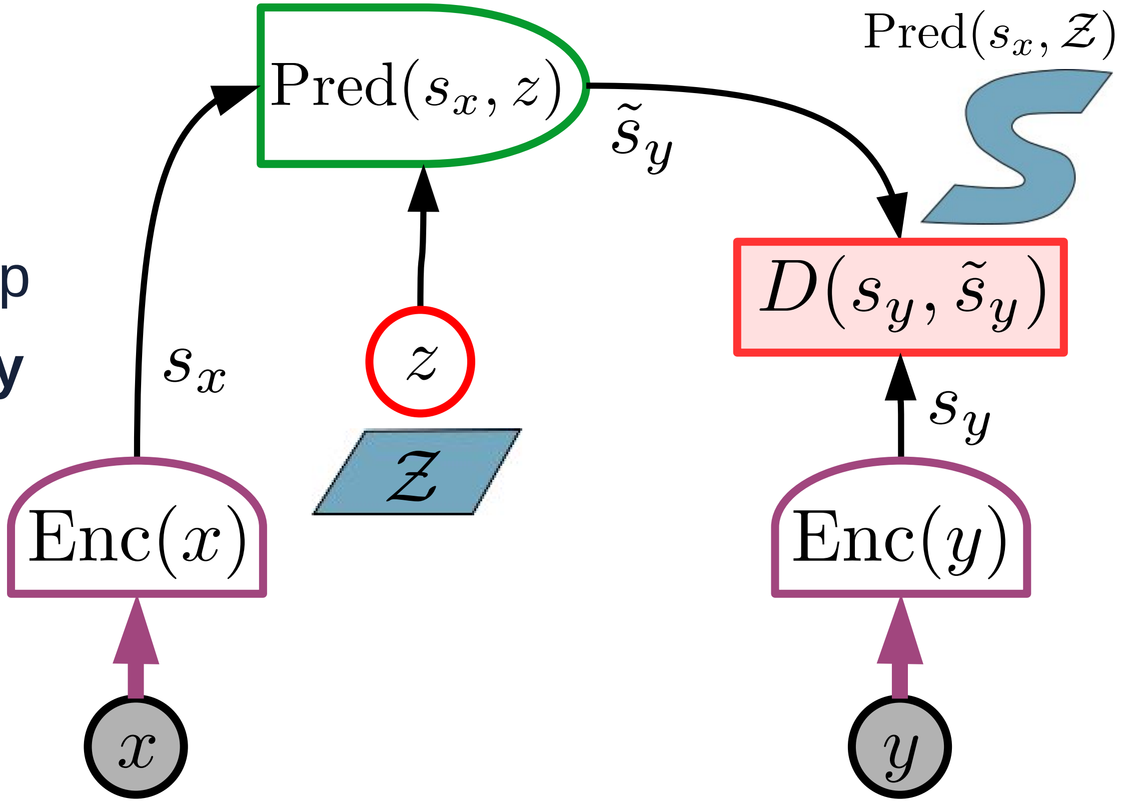




- **Multimodality through invariance properties of the right branch.**
  - Multiple y will produce the same representation and will have the same energy.

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.**



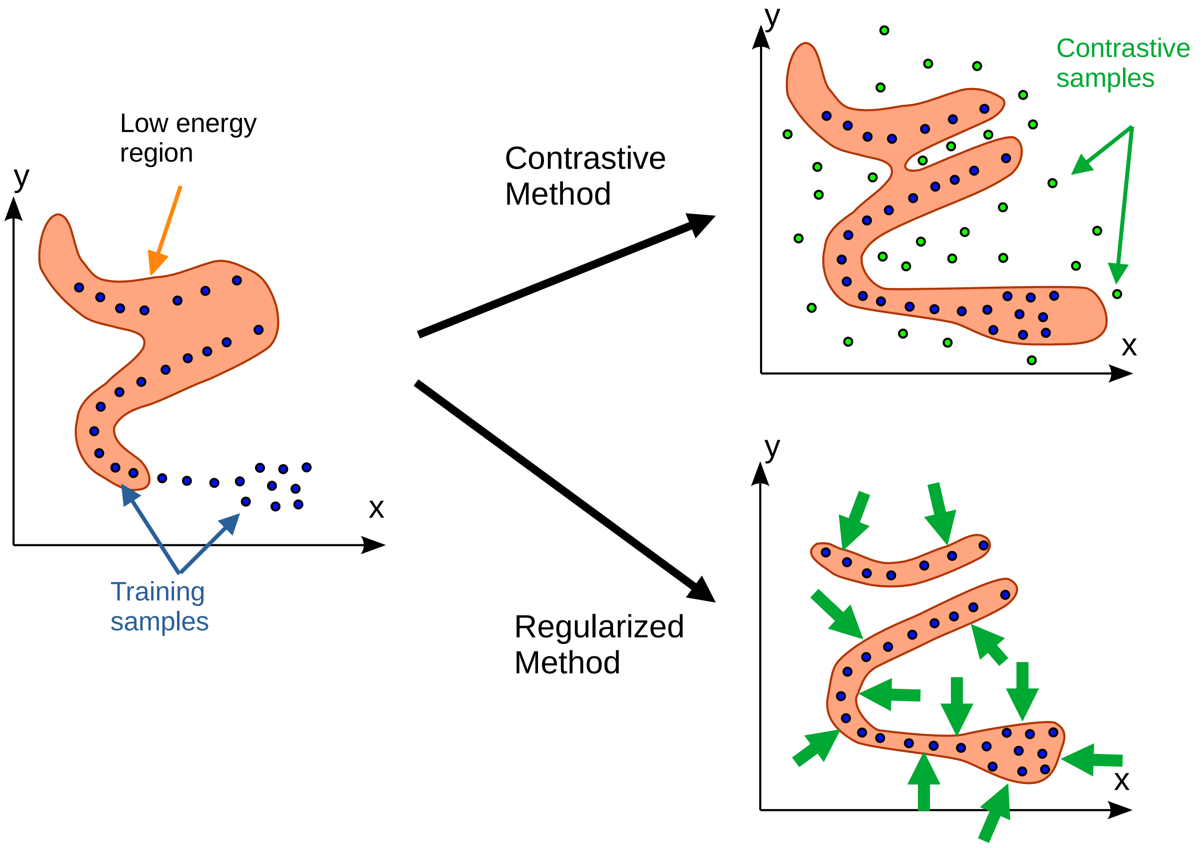
FACEBOOK AI

Contrastive Methods for joint embedding architectures

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

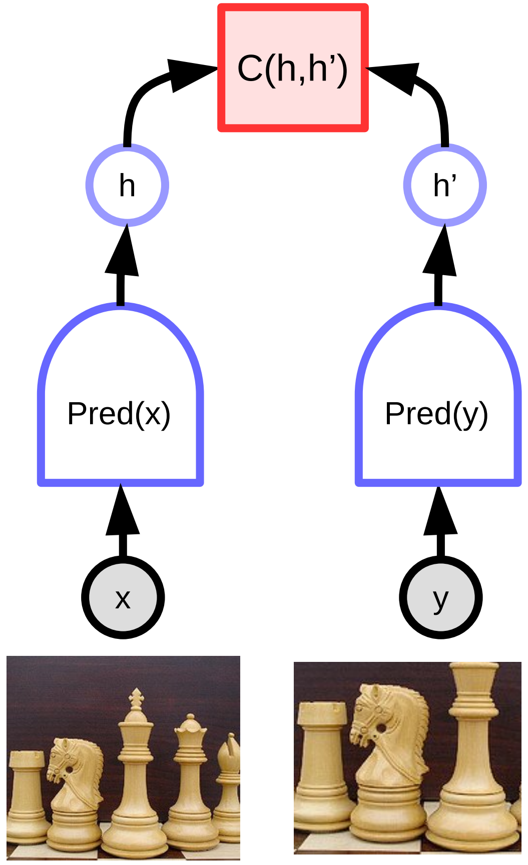
EBM Training: Contrastive vs Regularized methods

- **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

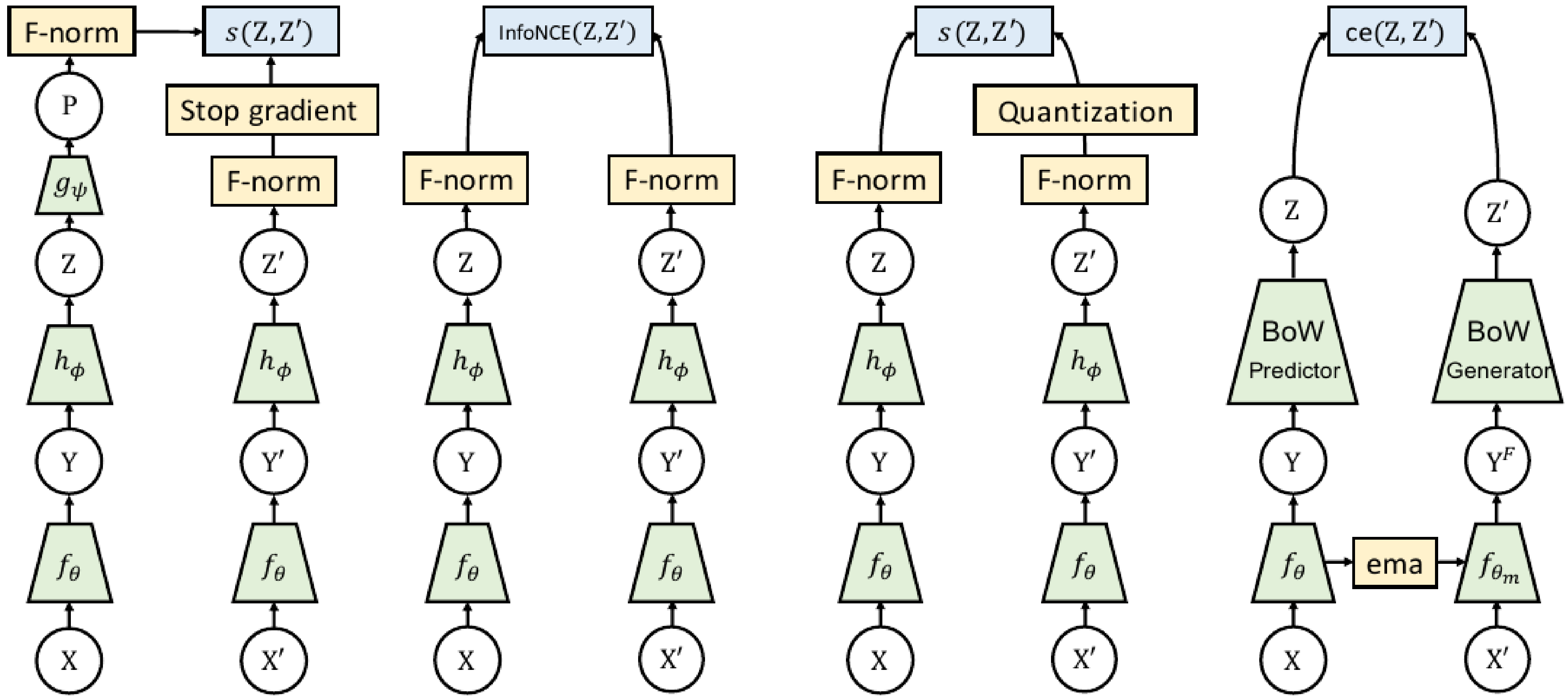


Joint Embedding Architectures

- **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

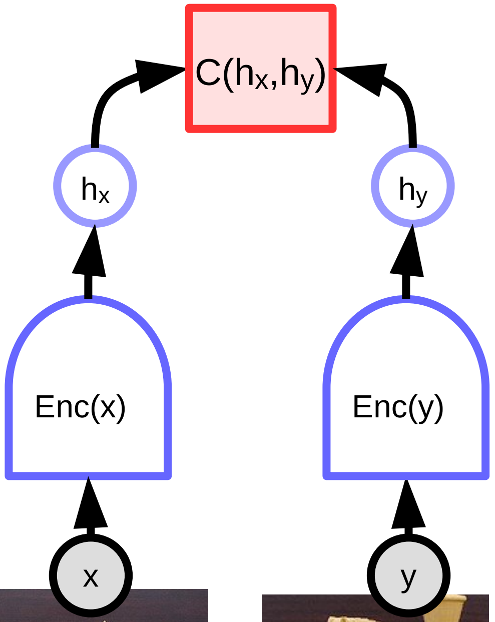
Contrastive Joint Embedding

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

- **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



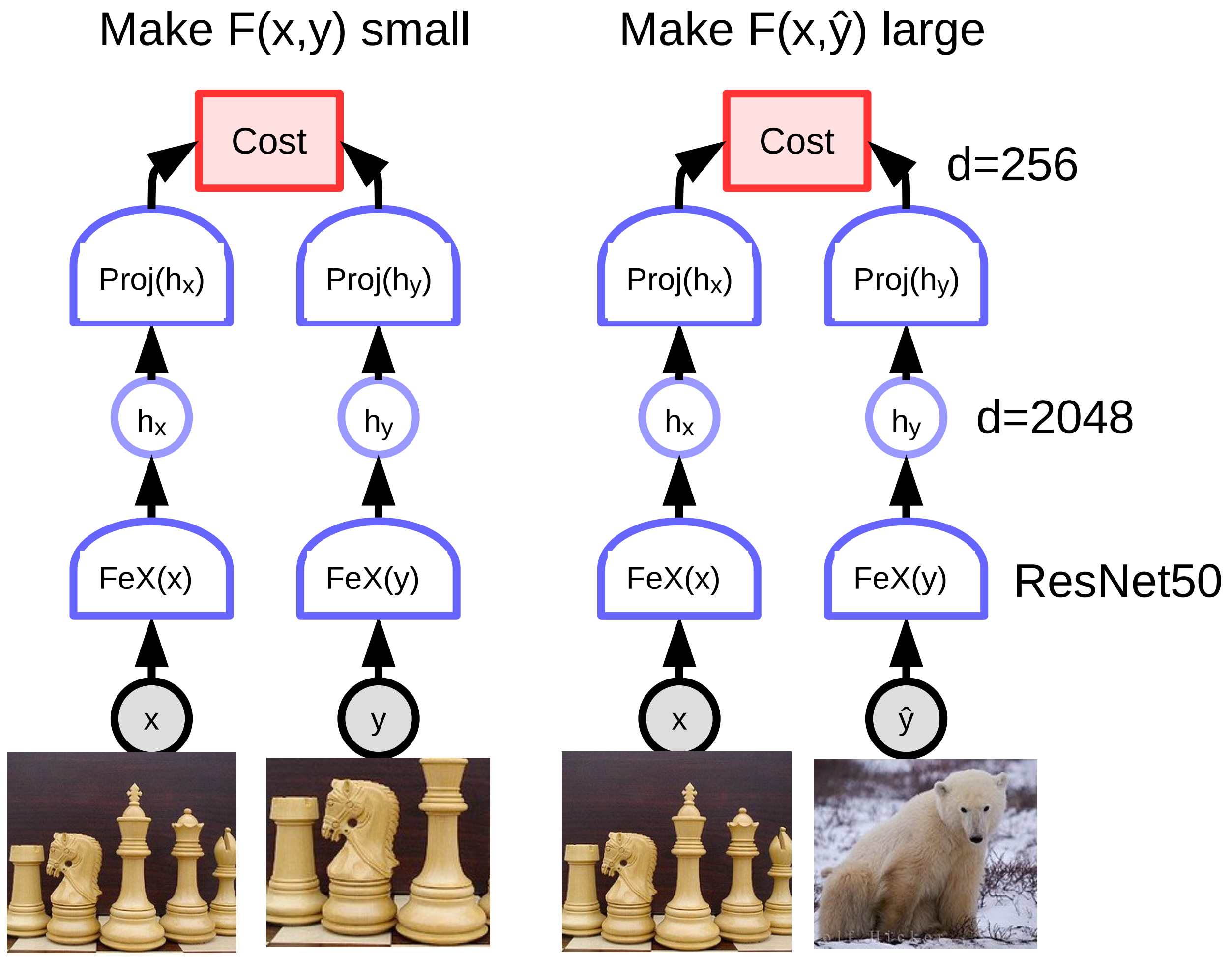
Positive pair:

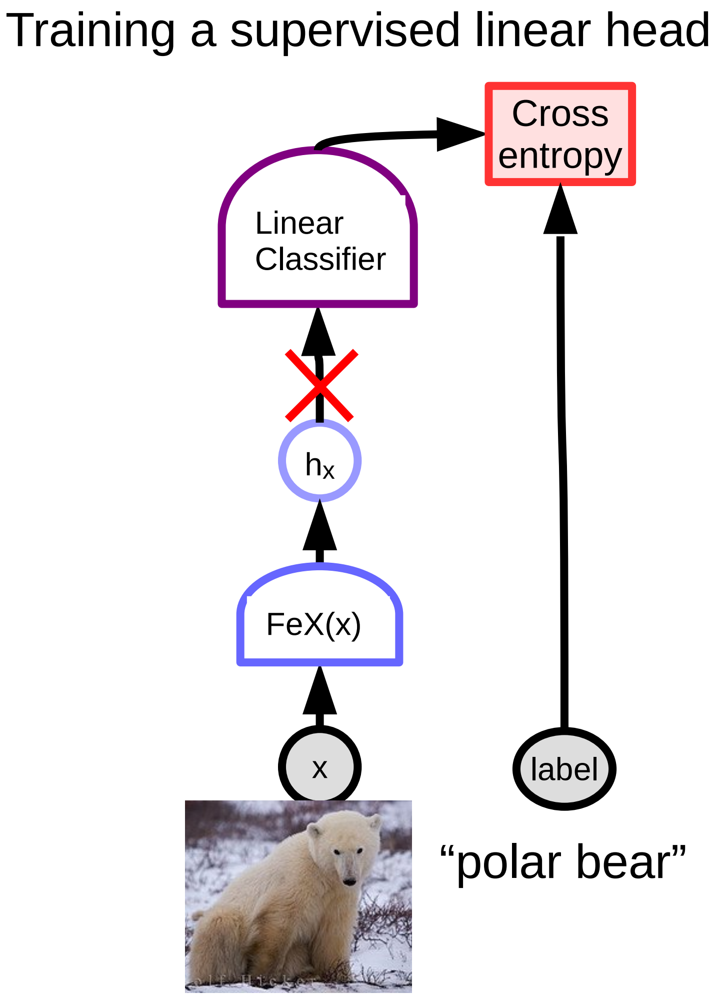
Make F small

Negative pair:

Make F large

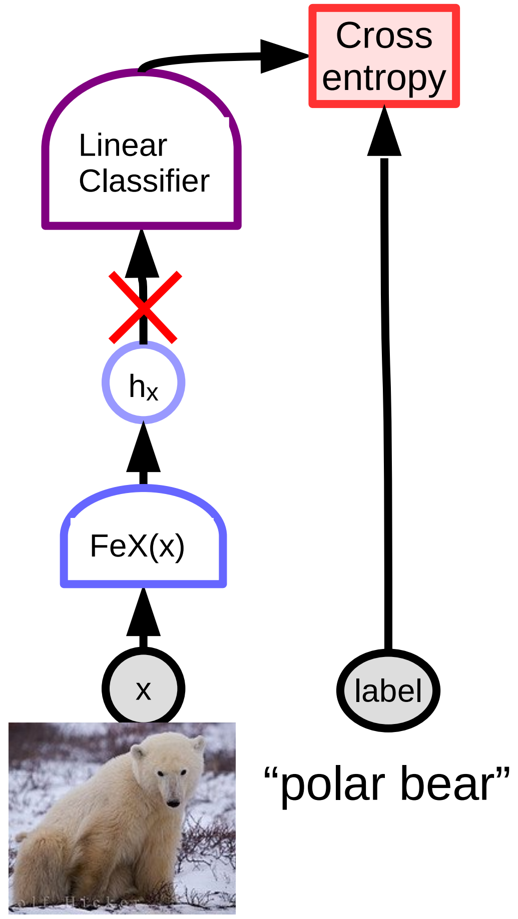
Contrastive Joint Embedding





Contrastive Joint Embedding

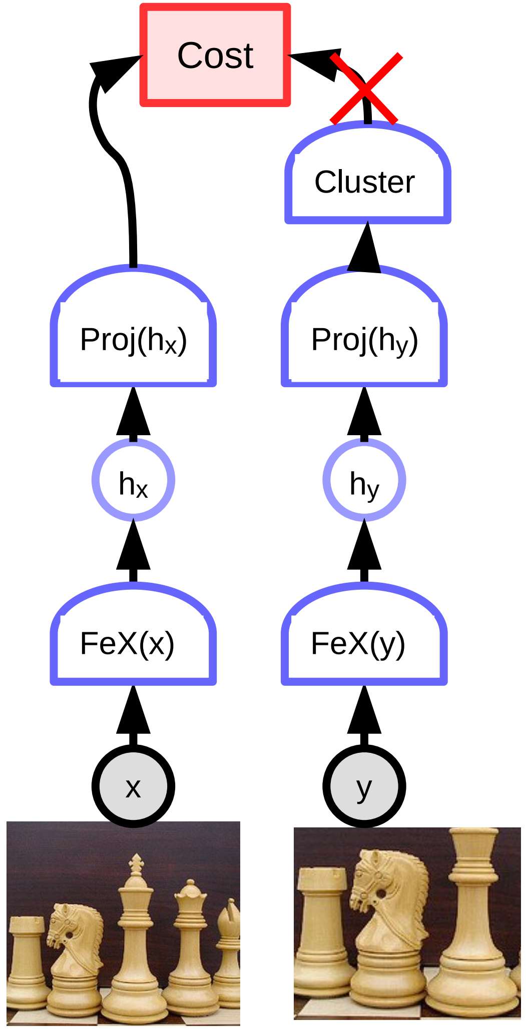
- **Issues:**
  - Hard negative mining
  - Expensive computationally
  - Only works for small dimension of embeddings (256)
- **Successful examples for image recognition:**
  - PIRL [Misra et al. Arxiv:1912.01991]
  - MoCo [He et al. Arxiv:1911.05722]
  - SimCLR [Chen et al. Arxiv:2002.05709]
- **Use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



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

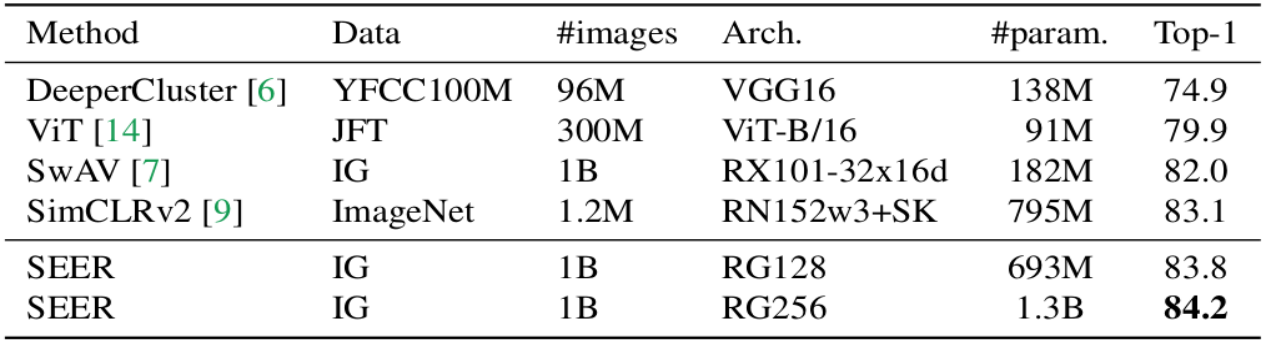
Quantization Methods

- **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

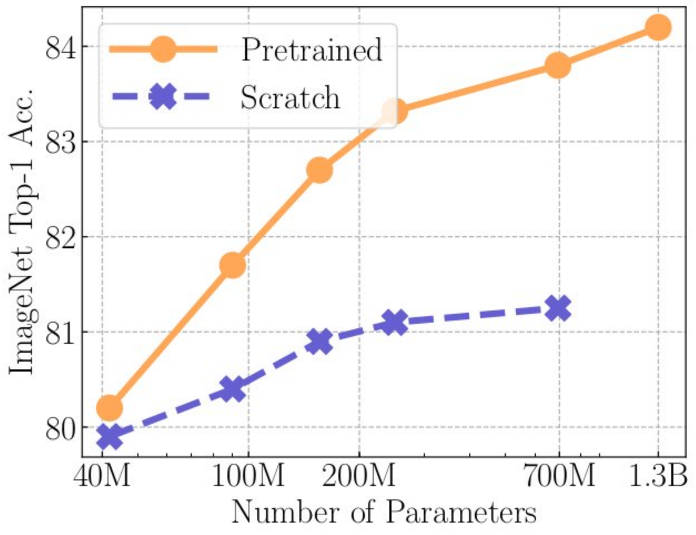


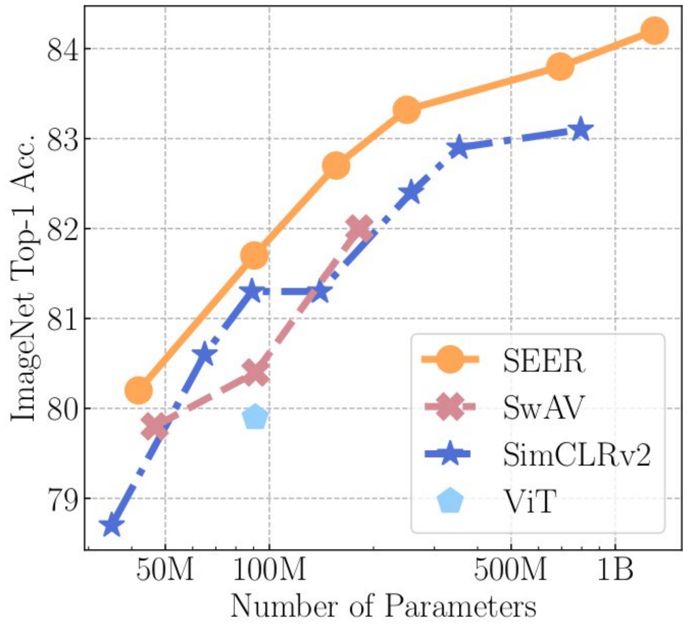
SEER [Goyal et al. ArXiv:2103.01988]

- **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/**



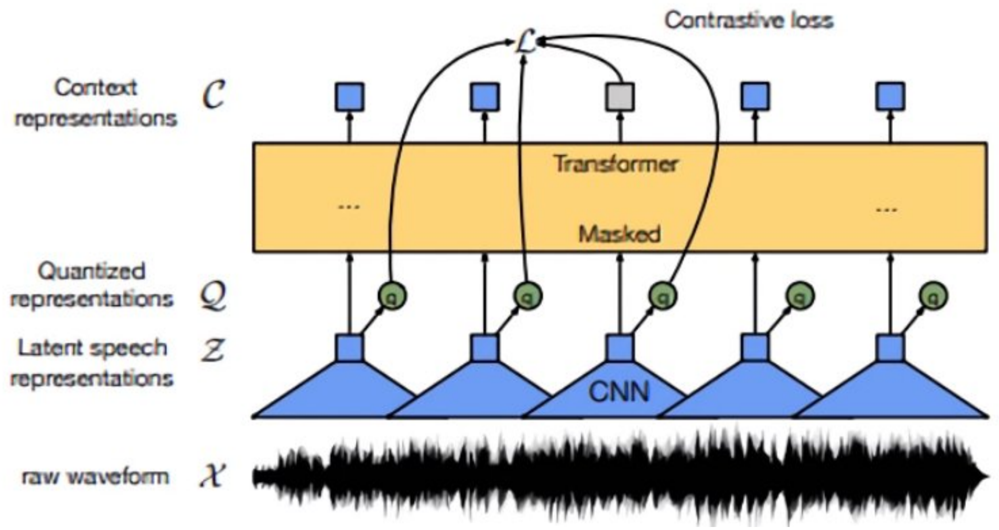
Method	Data	#images	Arch.	#param.	Top-1
DeeperCluster [6]	YFCC100M	96M	VGG16	138M	74.9
ViT [14]	JFT	300M	ViT-B/16	91M	79.9
SwAV [7]	IG	1B	RX101-32x16d	182M	82.0
SimCLRv2 [9]	ImageNet	1.2M	RN152w3+SK	795M	83.1
SEER	IG	1B	RG128	693M	83.8
SEER	IG	1B	RG256	1.3B	**84.2**

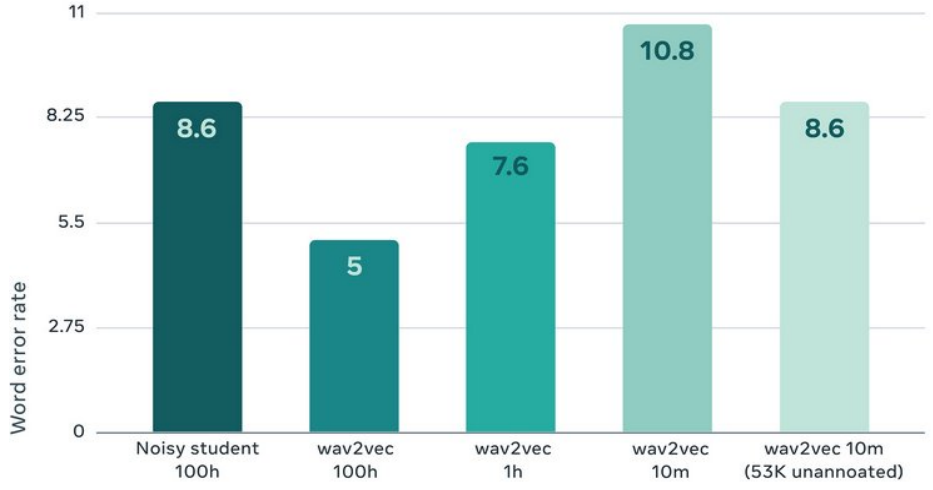




Wav2Vec 2.0: SSL for speech recognition

- **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

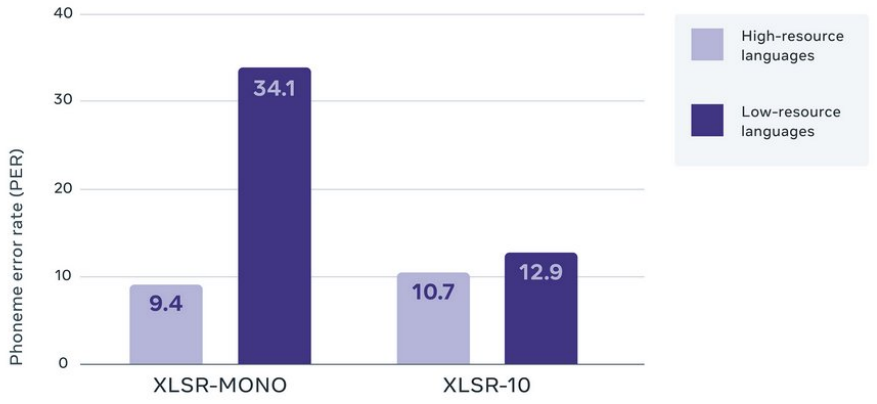

Contrastive loss
$\mathcal{L}$
Context representations $\mathcal{C}$
Transformer
...
Masked
Quantized representations $\mathcal{Q}$
q
Latent speech representations $\mathcal{Z}$
CNN
raw waveform $\mathcal{X}$



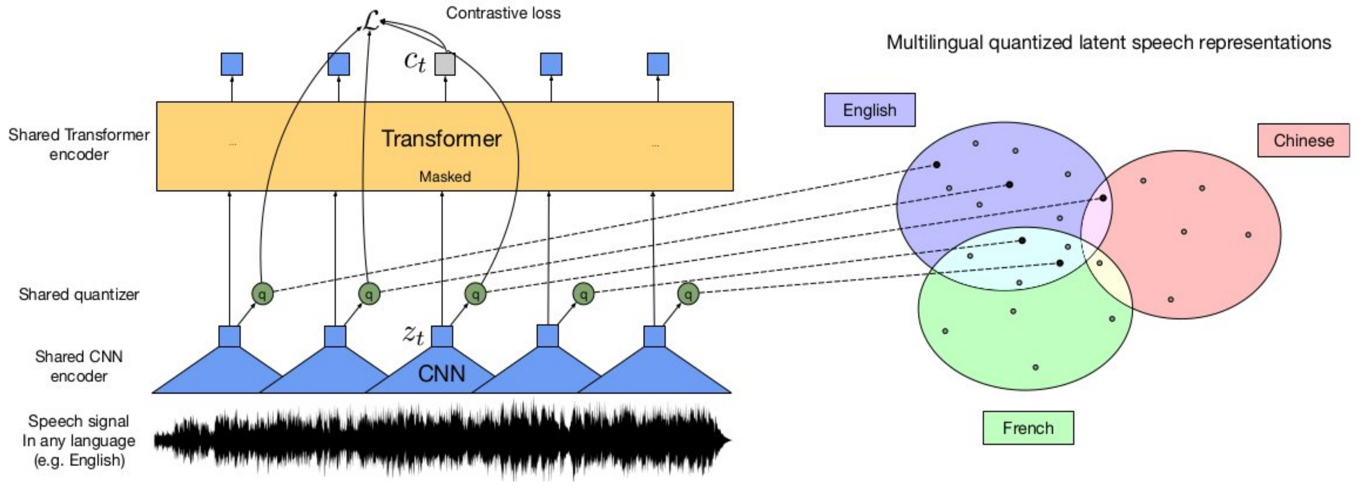
WER for Noisy Student self-training with 100 hours of labeled data. Wav2vec 2.0 with 100 hours, 1 hour, and only 10 minutes of labeled data. All models use the remainder of the LibriSpeech corpus (total 960 hours) as unannotated data, except for the last result, which uses 53K hours from LibriVox.

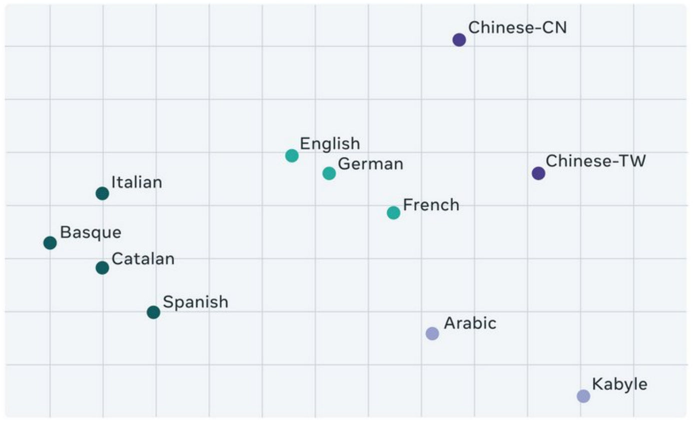
XLSR: multilingual speech recognition

- **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).


Contrastive loss
$\mathcal{L}$
c_t
Shared Transformer encoder
Transformer
Masked
Shared quantizer
q
z_t
Shared CNN encoder
CNN
Speech signal In any language (e.g. English)
Multilingual quantized latent speech representations
English
Chinese
French



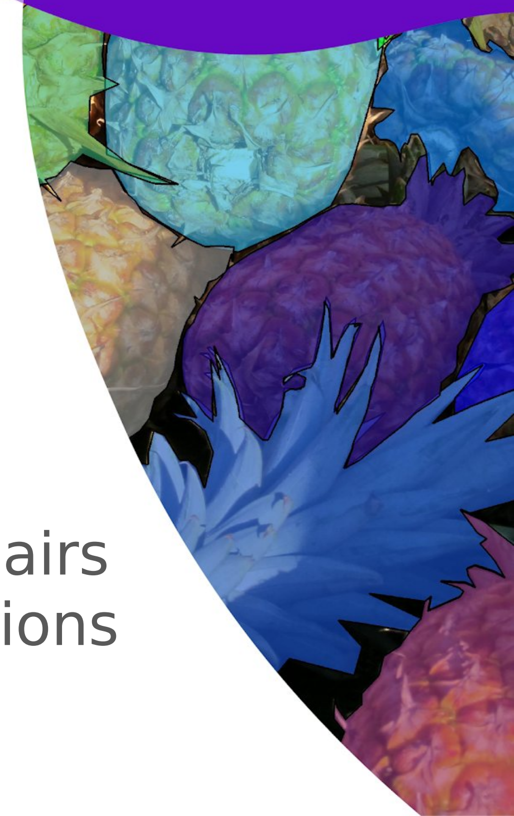
Visualization of how the learned units are used across languages. Graph shows a 2D PCA plot of how units are used in each language. Languages closer to each other, like English and German or Basque and Catalan, tend to use similar units.

Regularized Methods
for joint embedding architectures

This is the cool stuff!

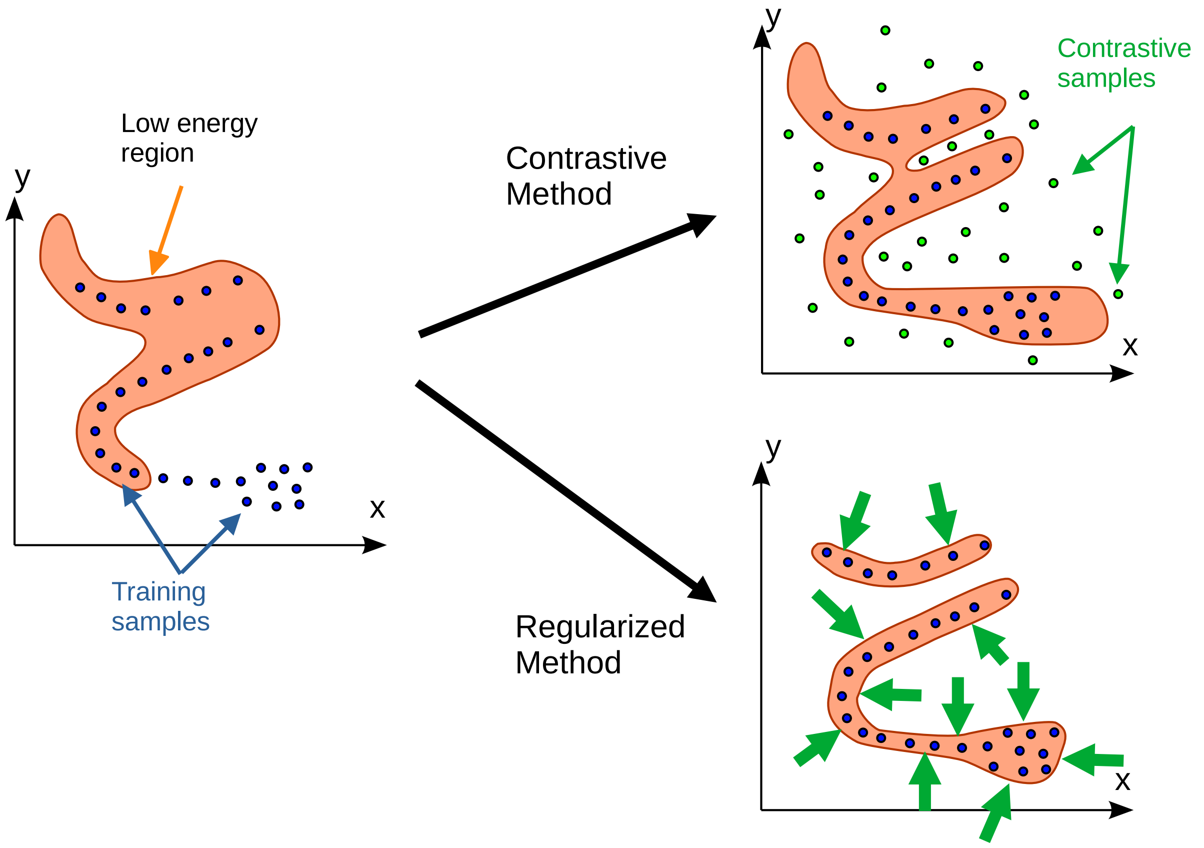
Push down on the energy of compatible sample pairs

Maximize the information capacity of representations



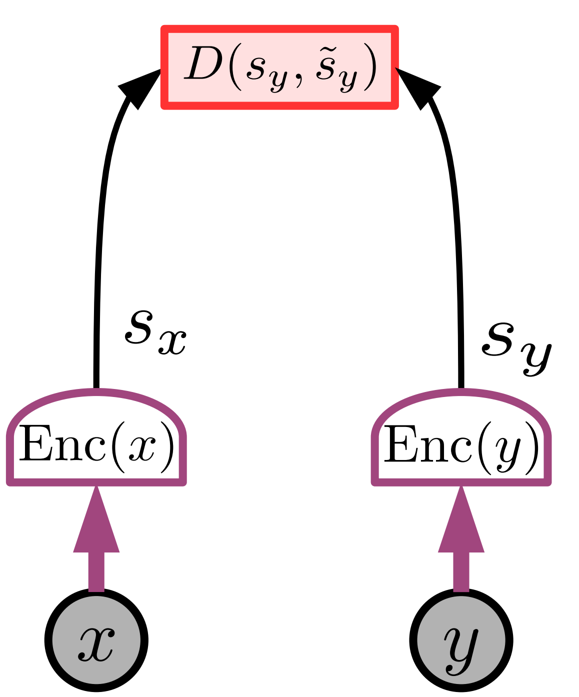
EBM Training: Contrastive vs Regularized methods

- **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

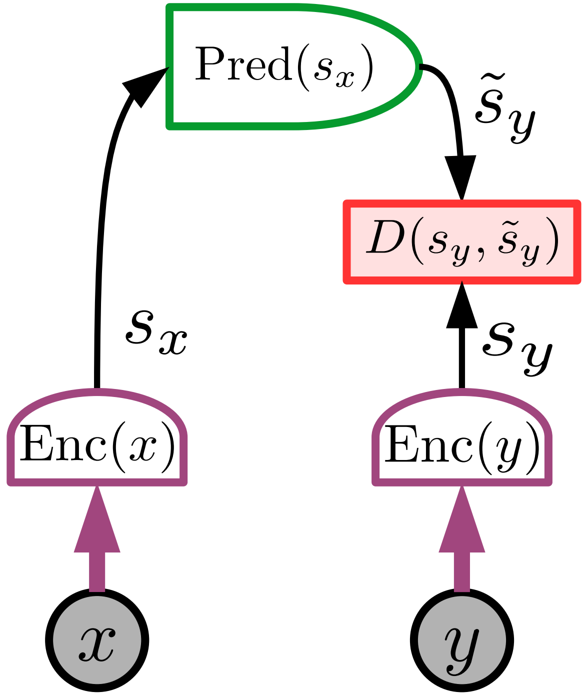


Joint Embedding Architecture (JEA)

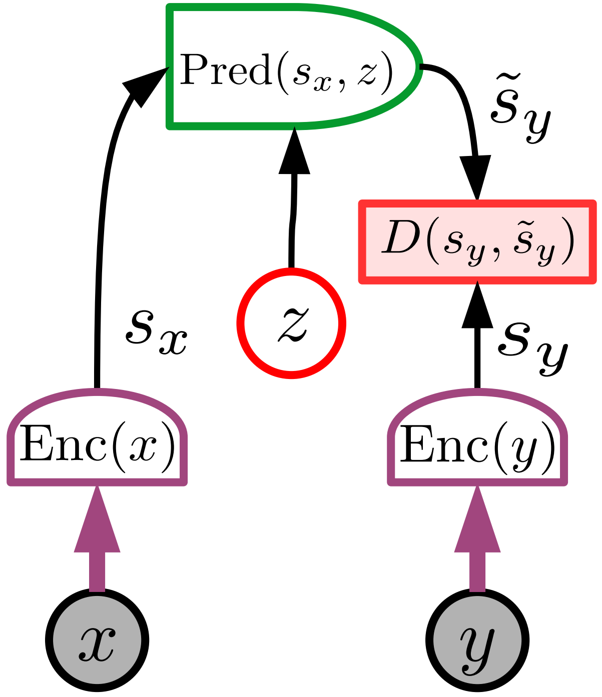
- **Computes abstract representations for x and y**
- **Makes predictions in representation space**



a) Joint Embedding Architecture (JEA)



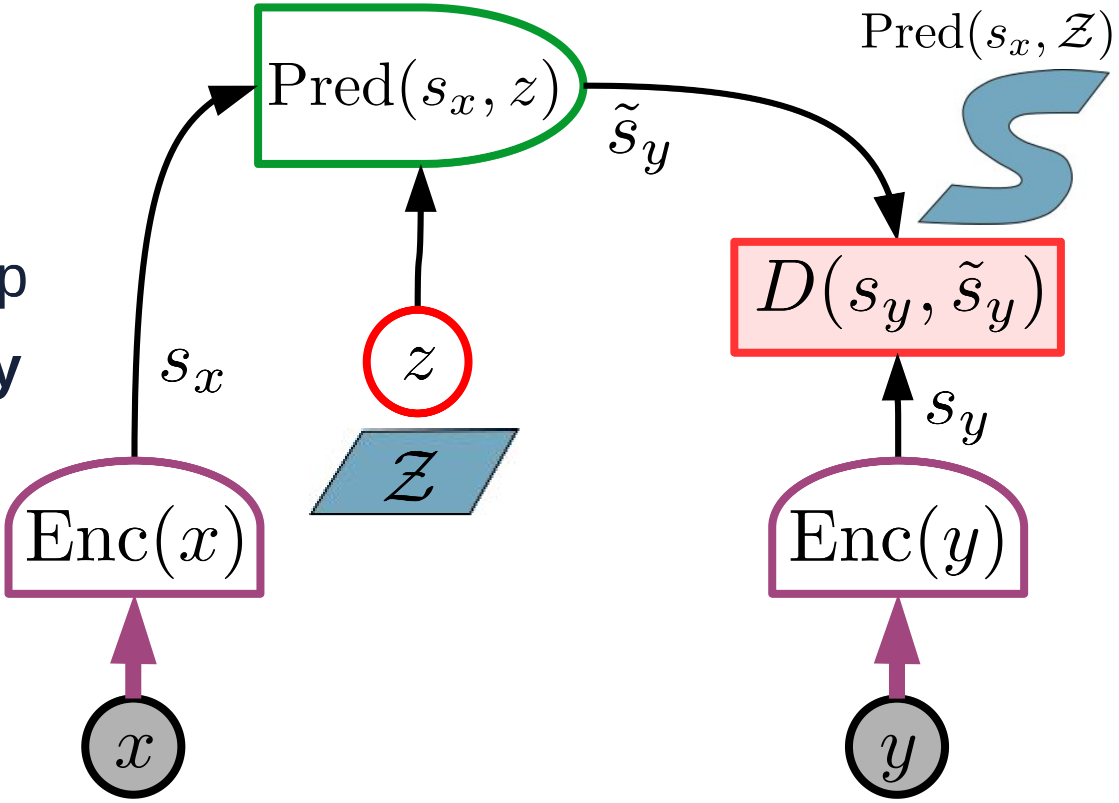
b) Deterministic Joint Embedding Predictive Architecture (DJEPA)



c) Joint Embedding Predictive Architecture (JEPA)

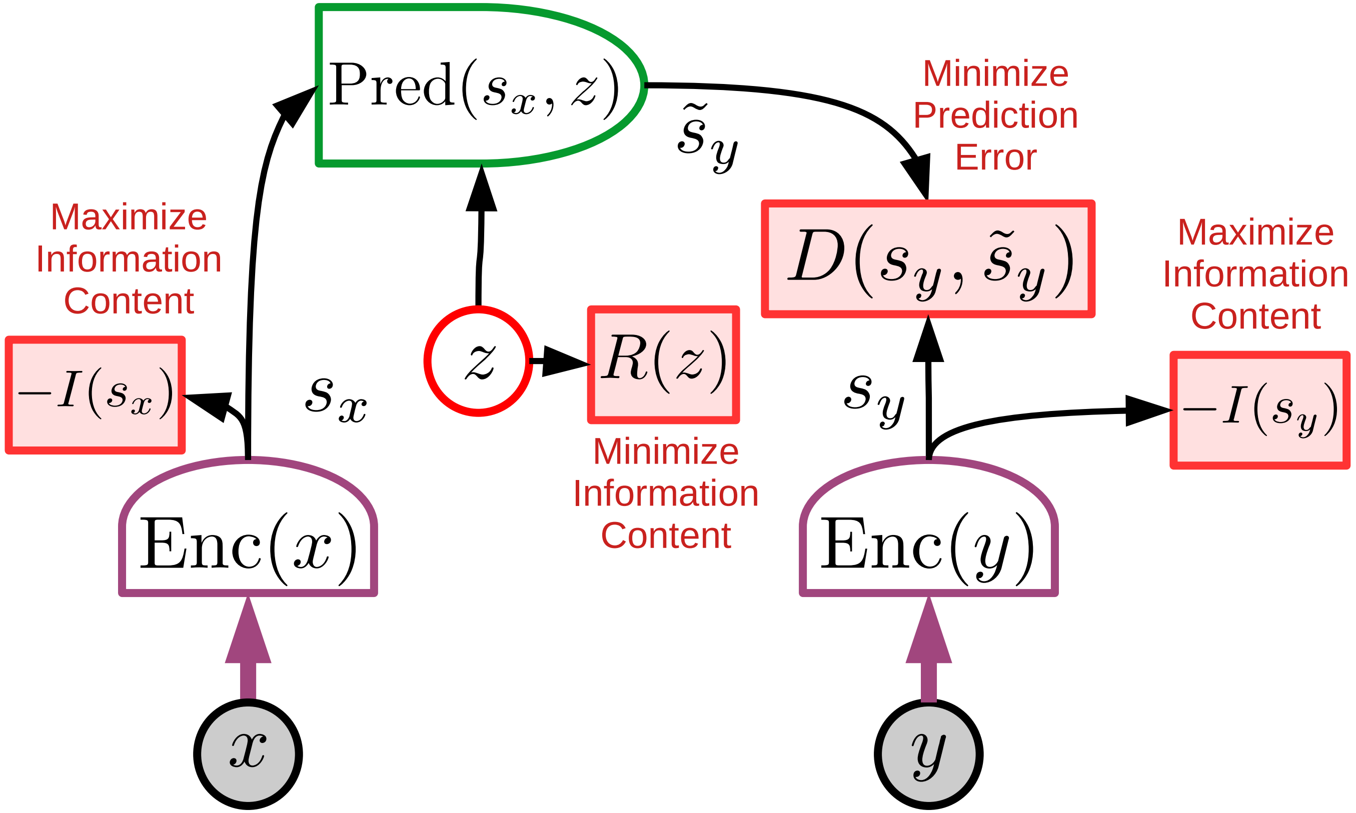
Joint Embedding Predictive Architecture (JEPA)

- **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.**

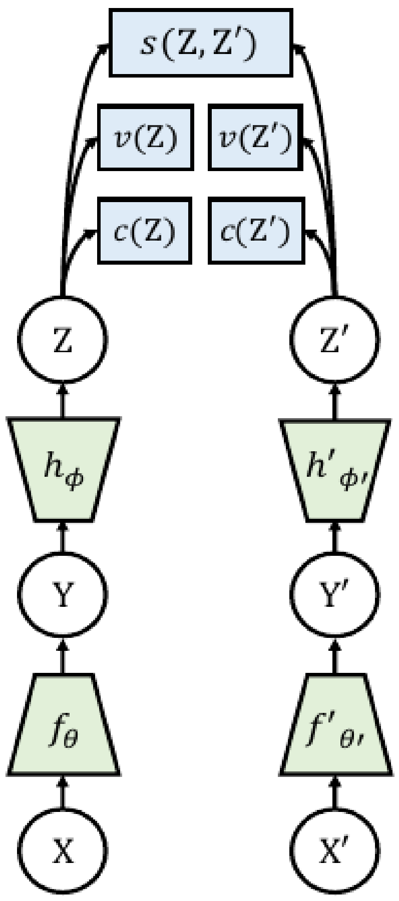


Training a JEPA (non contrastively)

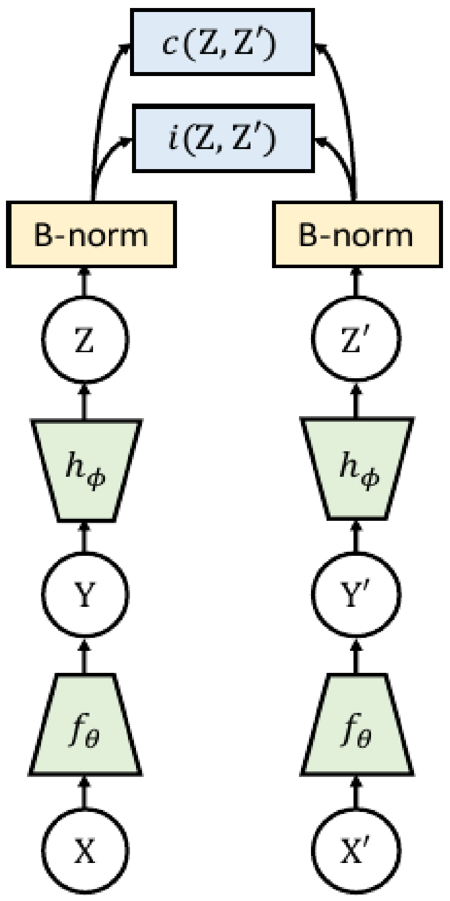
- **Four terms in the cost**
  - Maximize information content in representation of x
  - Maximize information content in representation of y
  - Minimize Prediction error
  - Minimize information content of latent variable z



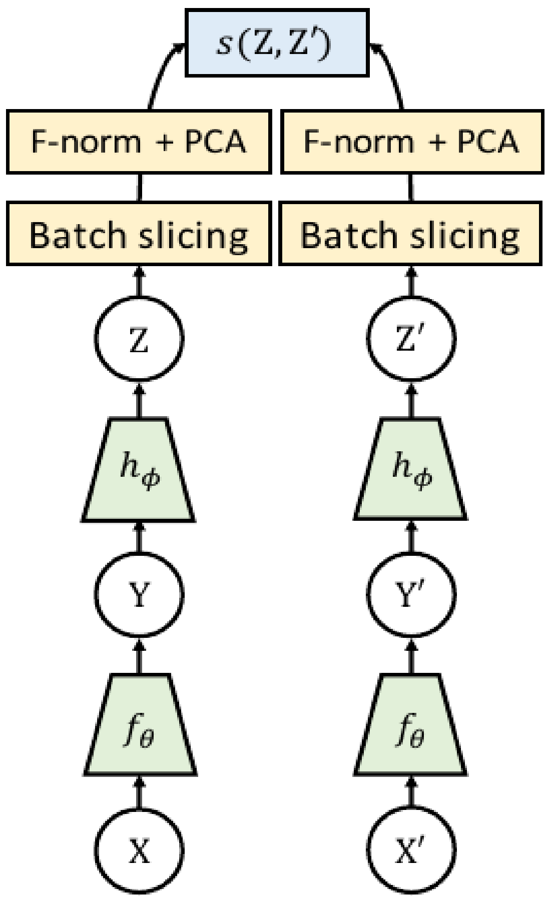
Regularized (Non-Contrastive) Joint Embedding Methods



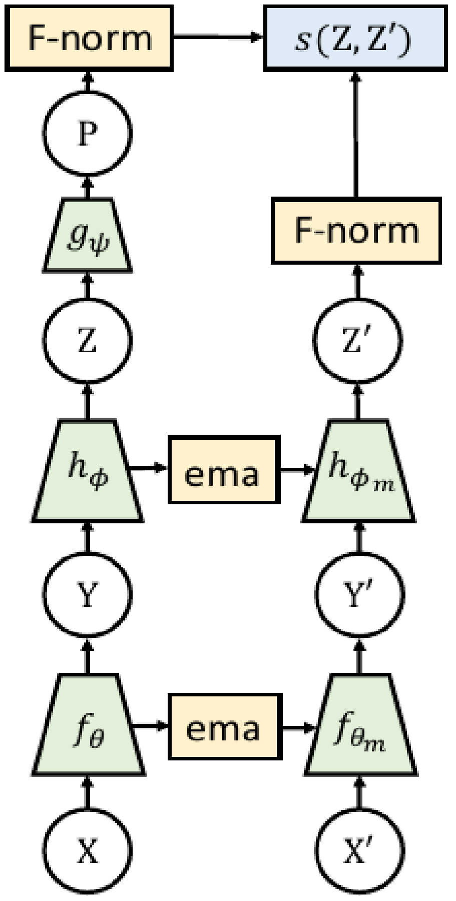
(a) VICReg



(b) Barlow Twins



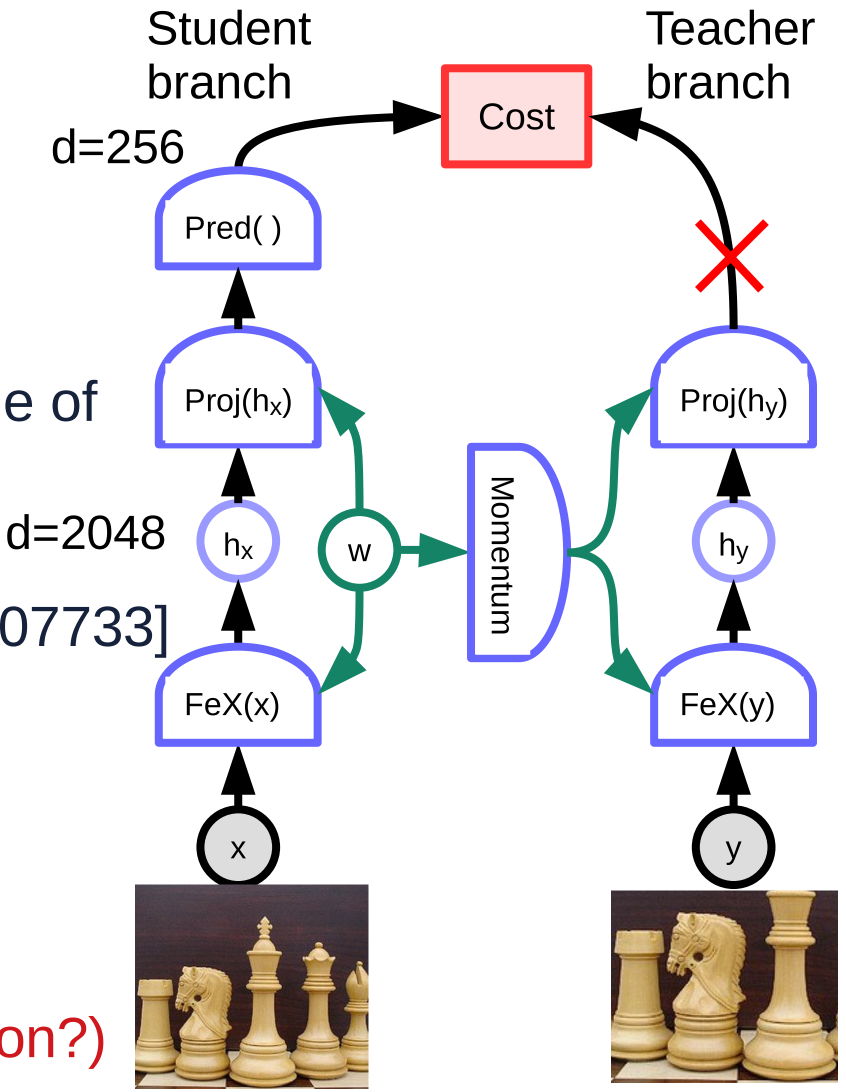
(c) W-MSE



(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?)

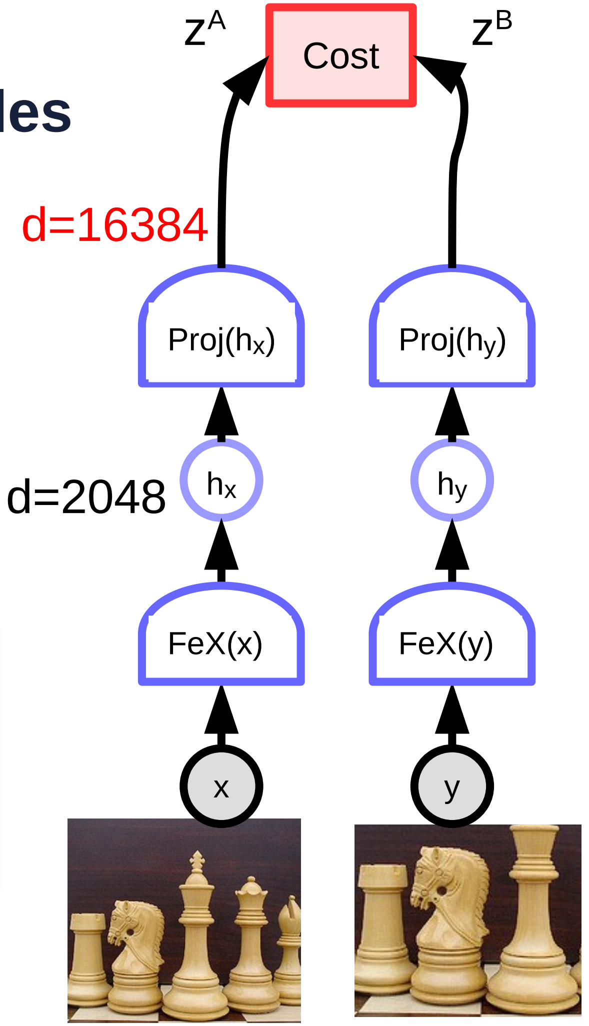


Information Maximization

- **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

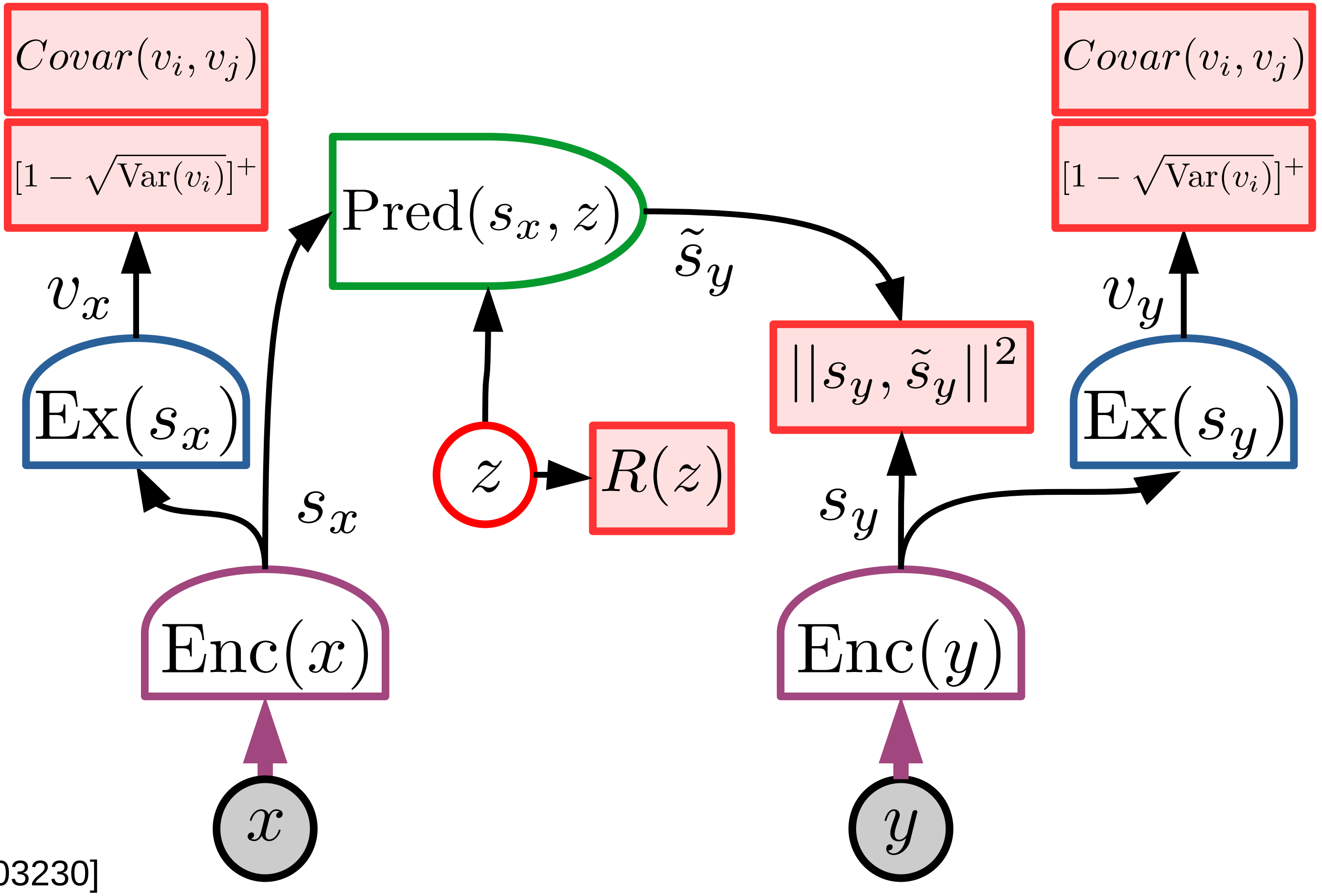
$$\mathcal{C}_{ij} \triangleq \frac{\sum_b z^A_{b,i} z^B_{b,j}}{\sqrt{\sum_b \left(z^A_{b,i}\right)^2} \sqrt{\sum_b \left(z^B_{b,j}\right)^2}}$$

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



VICReg: Variance, Invariance, Covariance Regularization

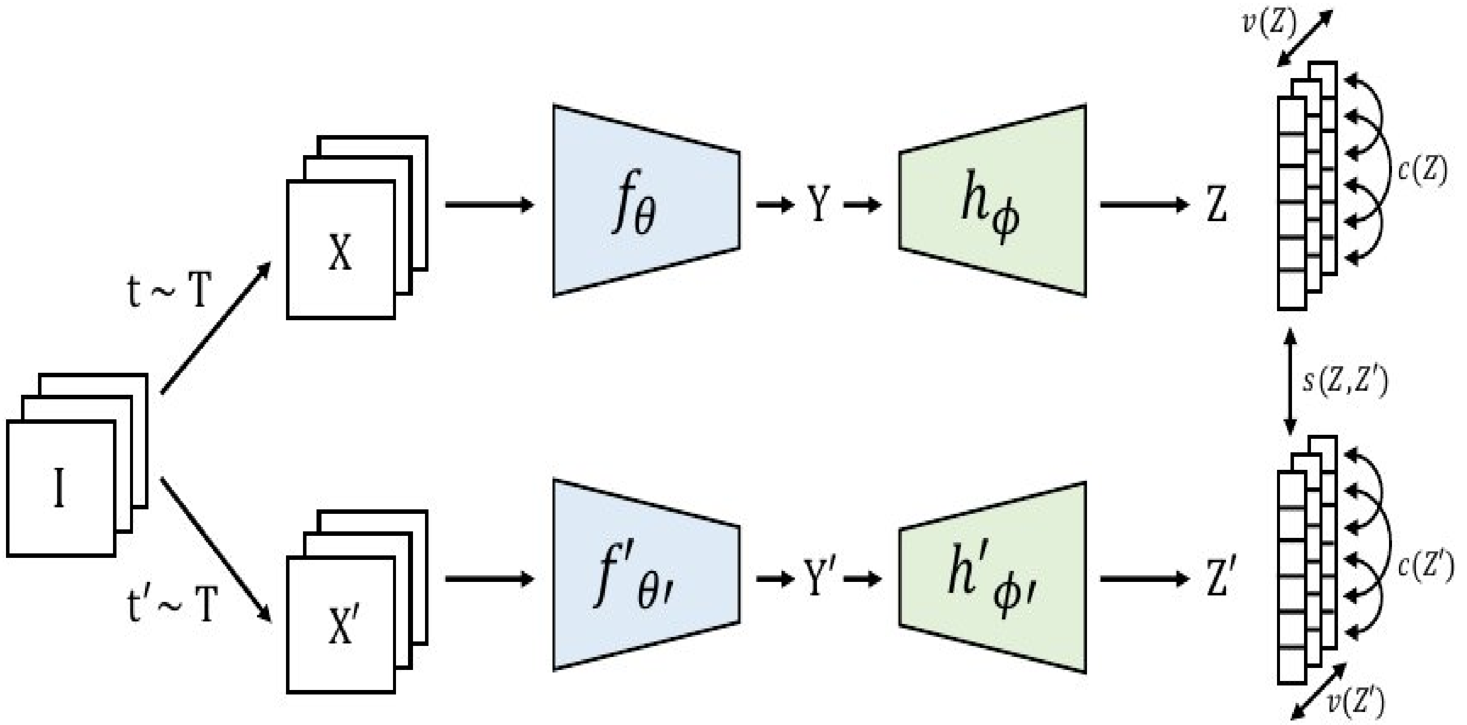
- **Variance:**
  - Maintains variance of components of representations
- **Covariance:**
  - Decorrelates components of covariance matrix of representations
- **Invariance:**
  - Minimizes prediction error.



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

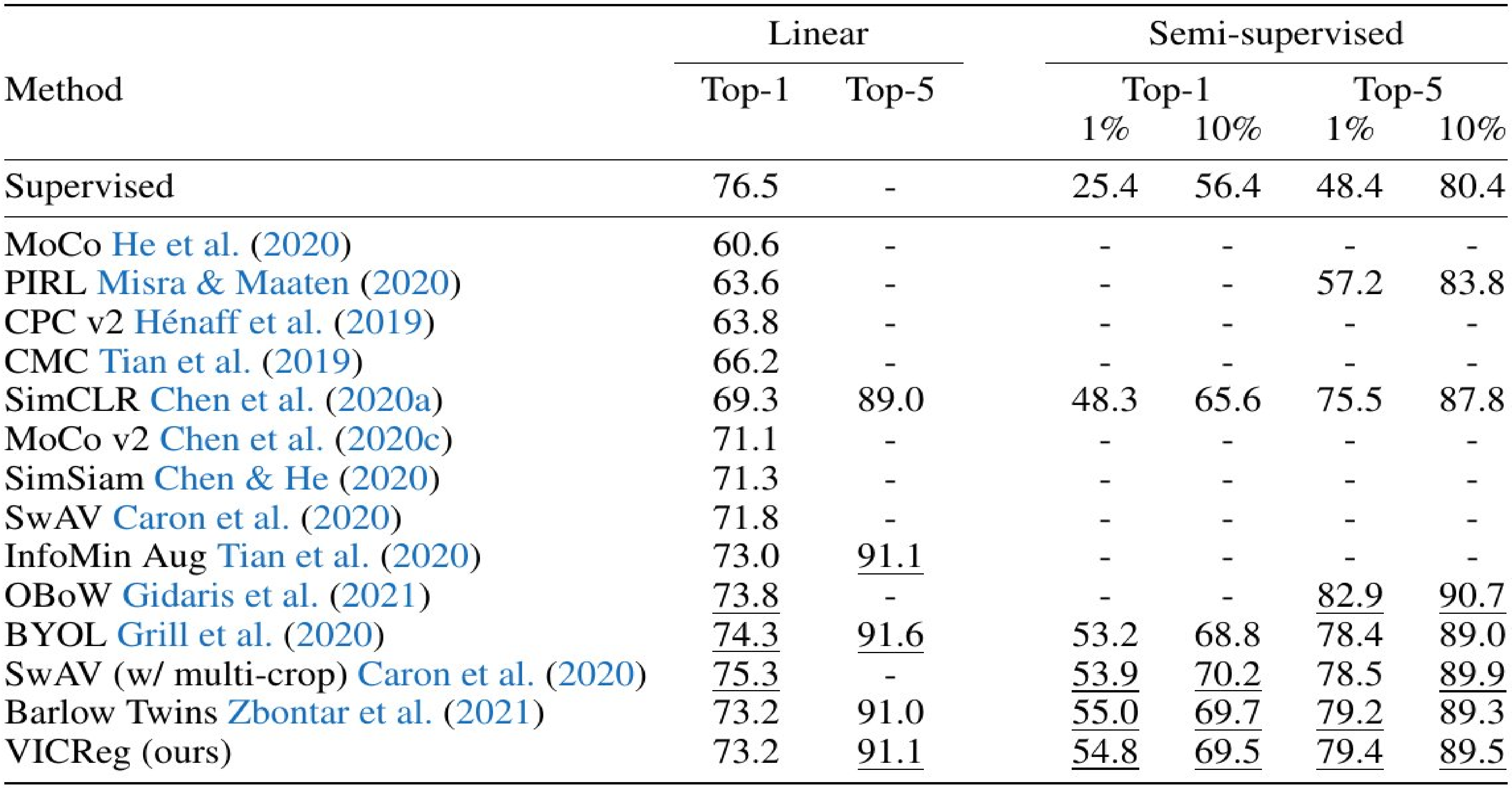
VICReg: Variance-Invariance-Covariance Regularization

- 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



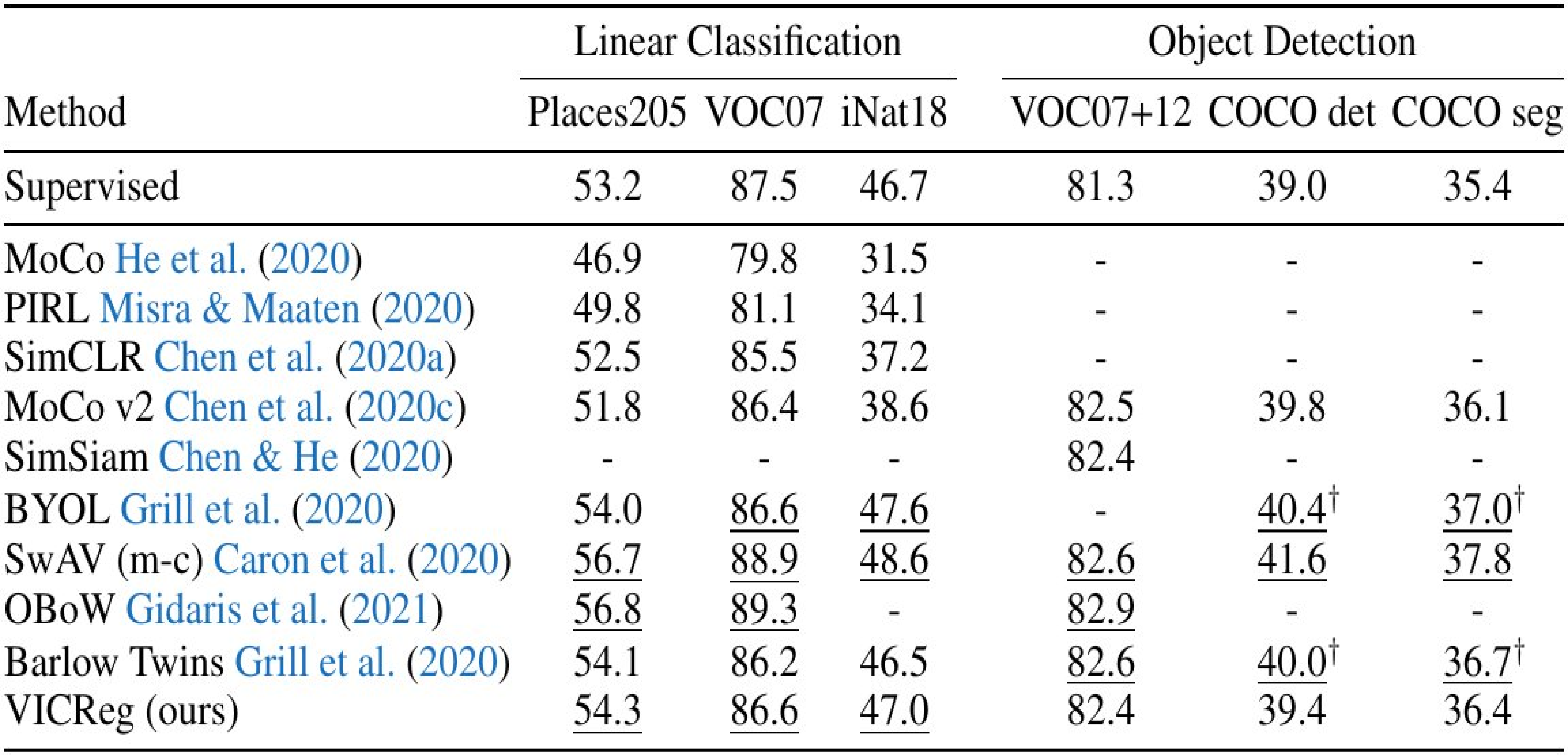
v	: maintain variance
c	: bring covariance to zero
s	: minimize distance
T	: distribution of transformations
t,t’	: random transformations
$f_\theta, f'_{\theta'}$	: encoders
$h_\phi, h'_{\phi'}$	: expanders
I	: batch of images
X, X’	: batches of views
Y, Y’	: batches of representations
Z, Z’	: batches of embeddings

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.



	Linear Classification			Object Detection		
Method	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

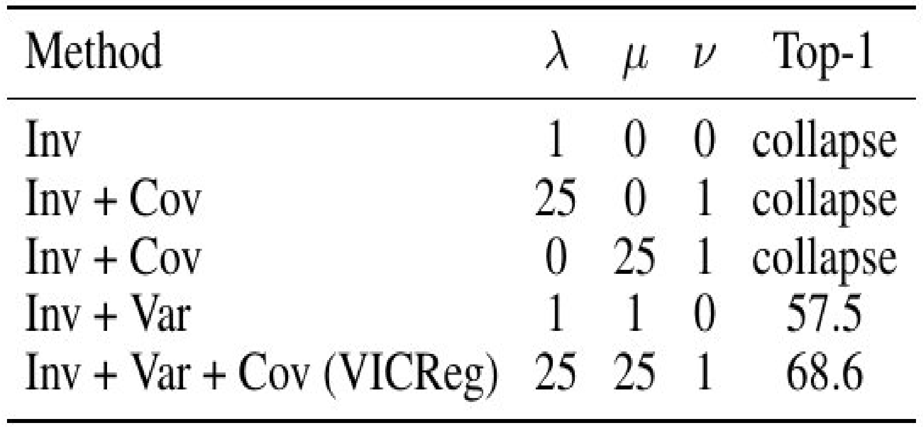
VICReg: no need for normalization, momentum encoder, predictor...

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

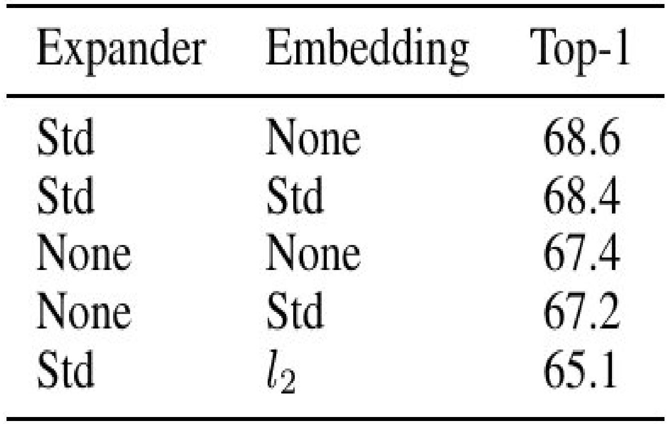
VICReg: Variance/Covariance regularization helps other methods

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**

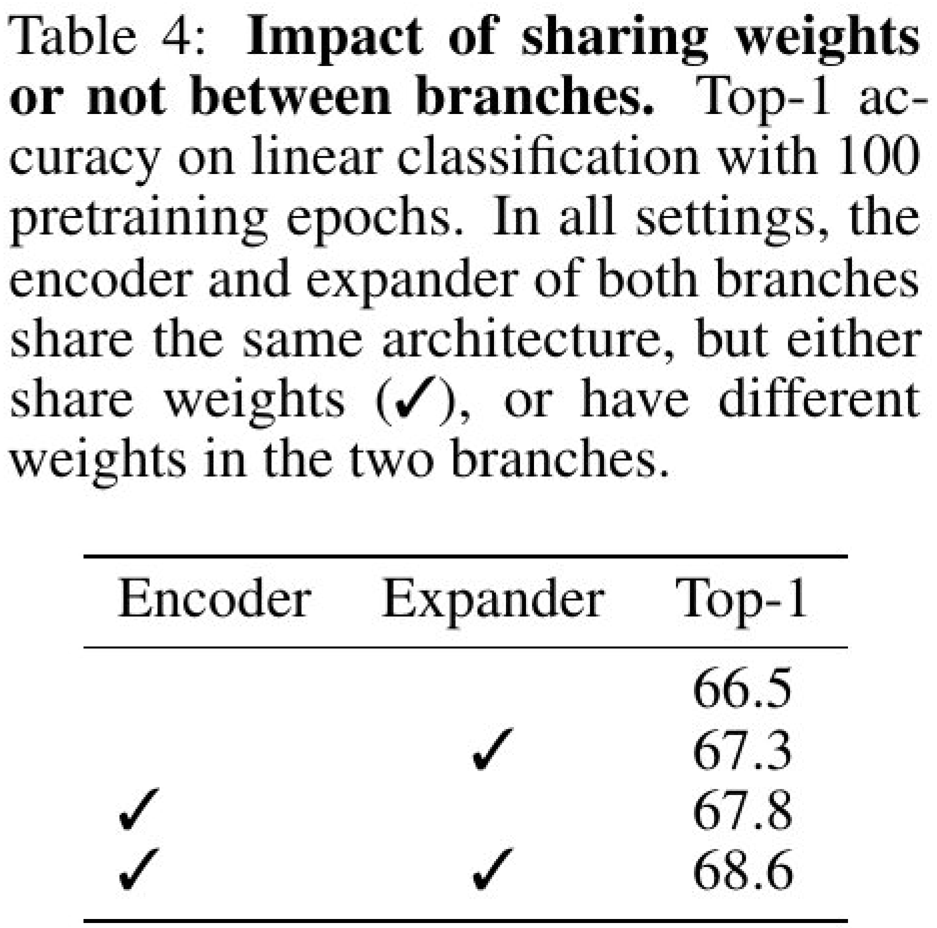
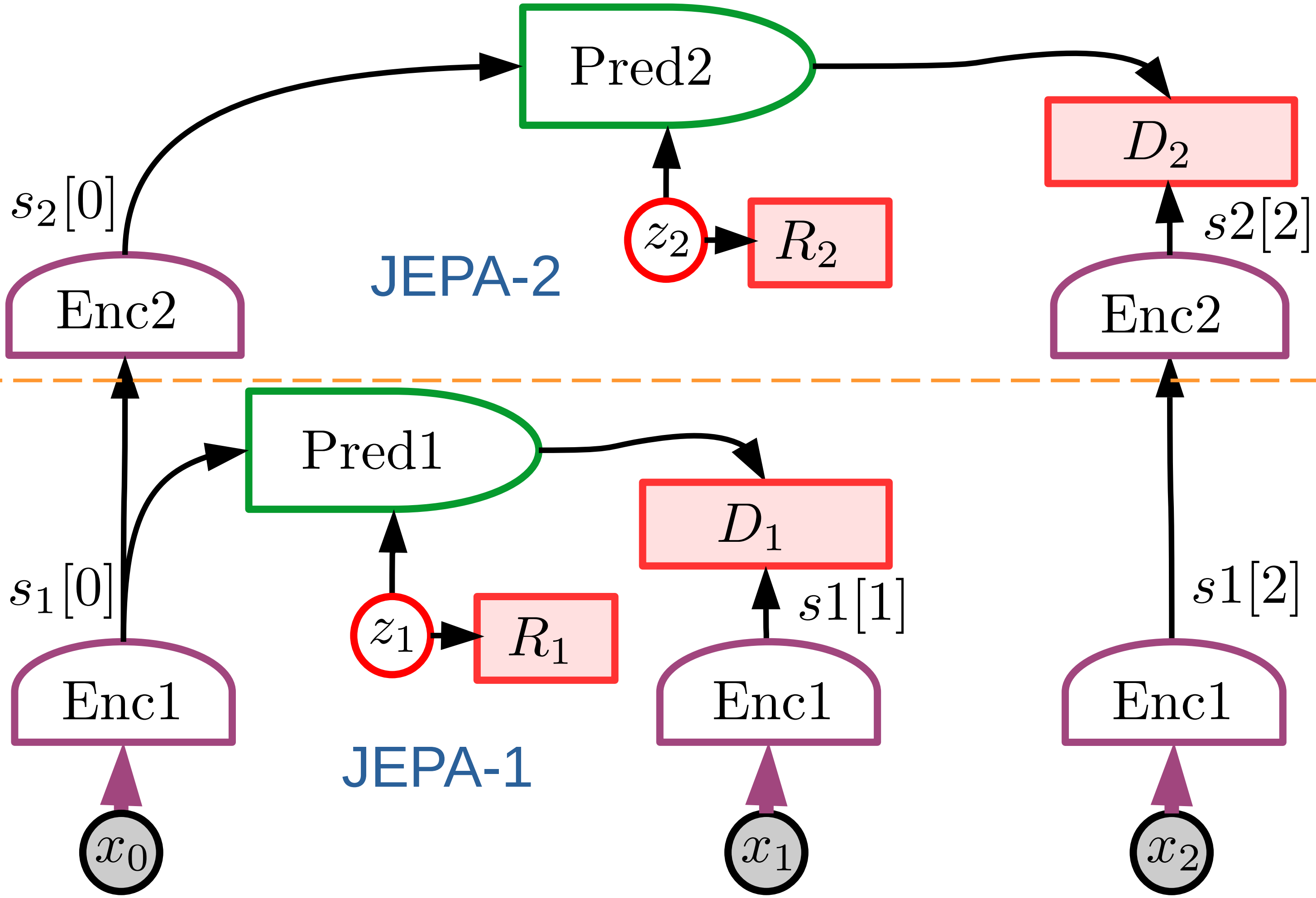


Table 4: **Impact of sharing weights or not between branches.** Top-1 accuracy on linear classification with 100 pretraining epochs. In all settings, the encoder and expander of both branches share the same architecture, but either share weights (✓), or have different weights in the two branches.

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

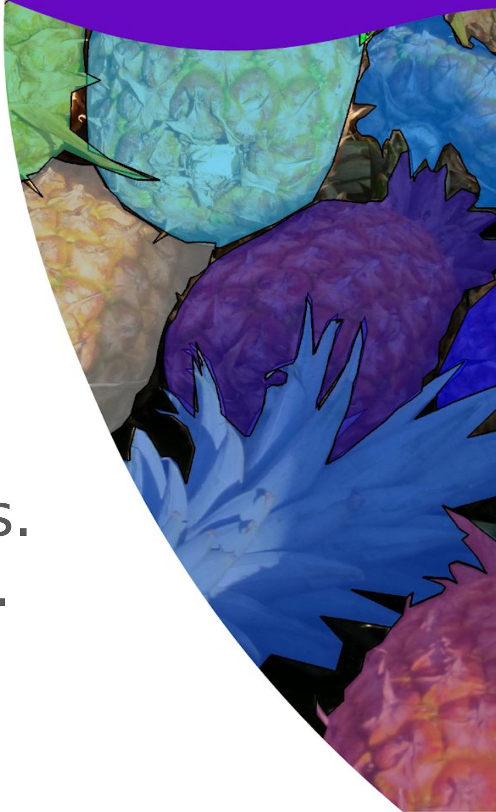


JEPA and H-JEPA

- **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

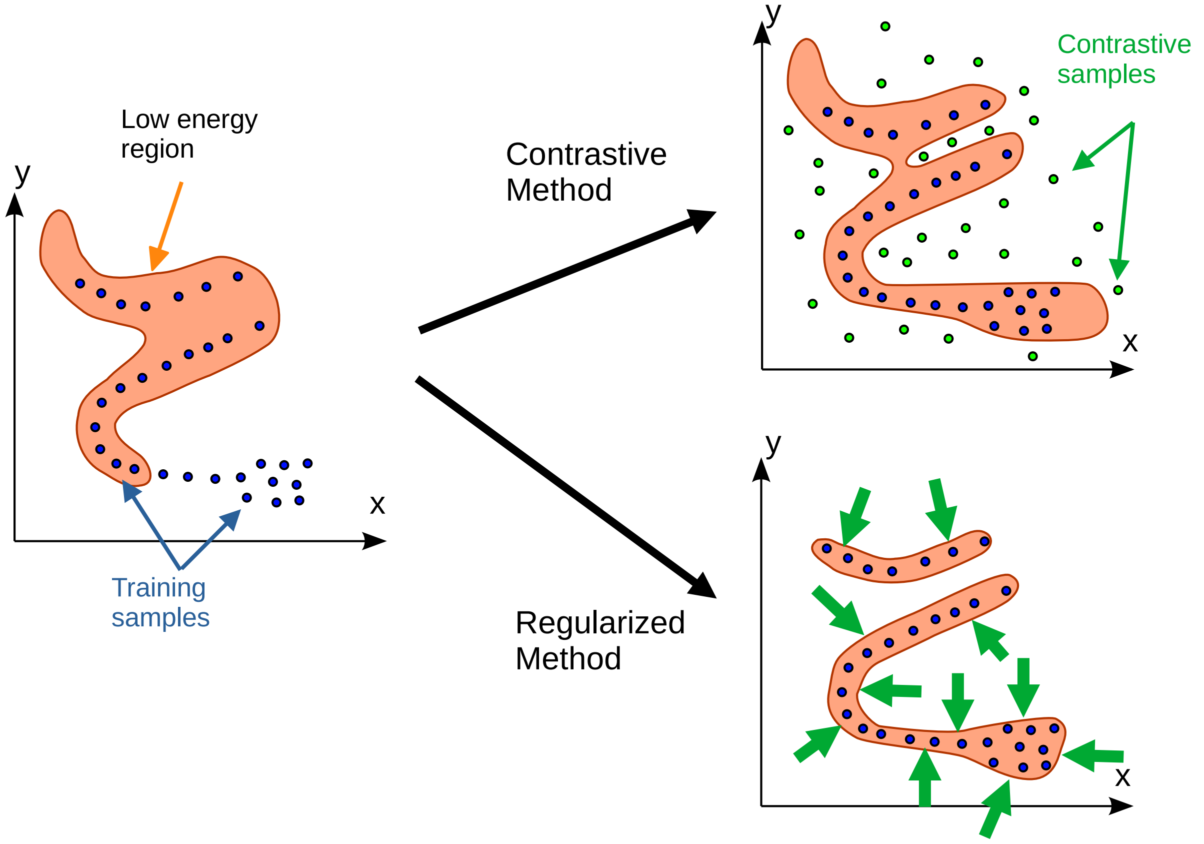
Push down on the energy of training samples.
Minimize the capacity of the latent variables.



Y. LeCun

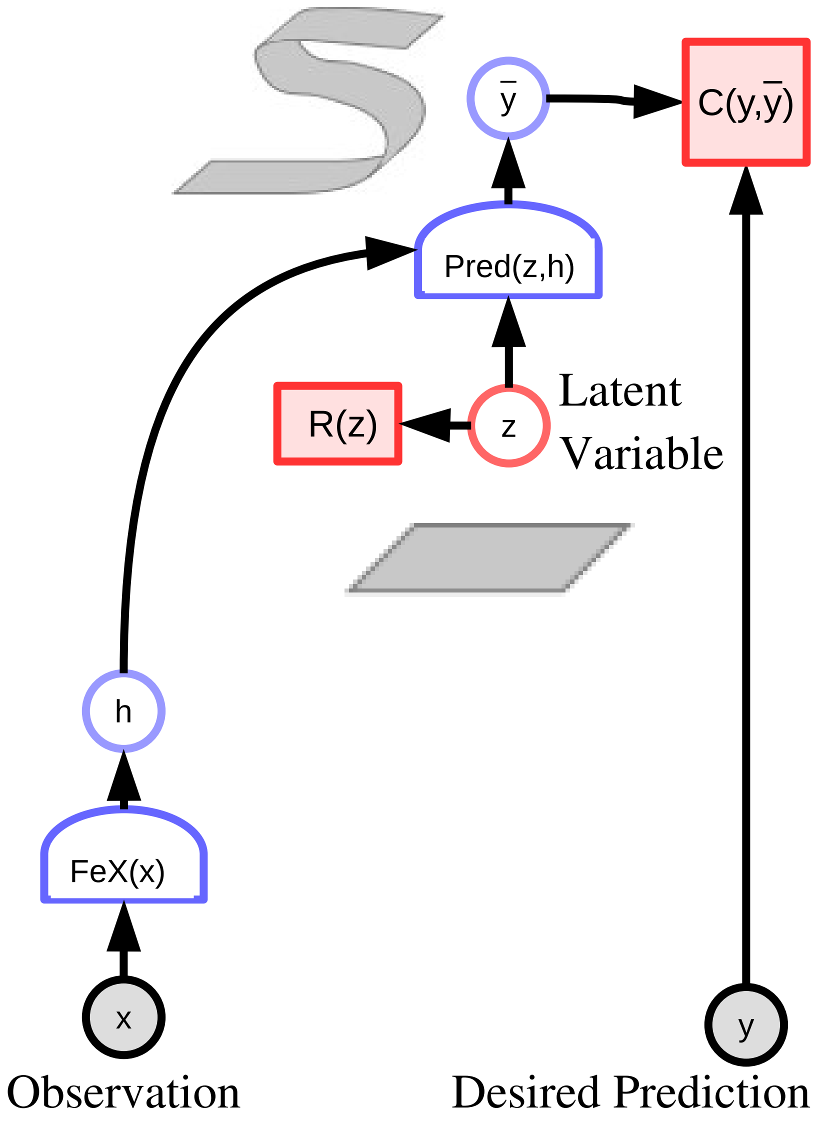
EBM Training: Contrastive vs Regularized methods

- **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.

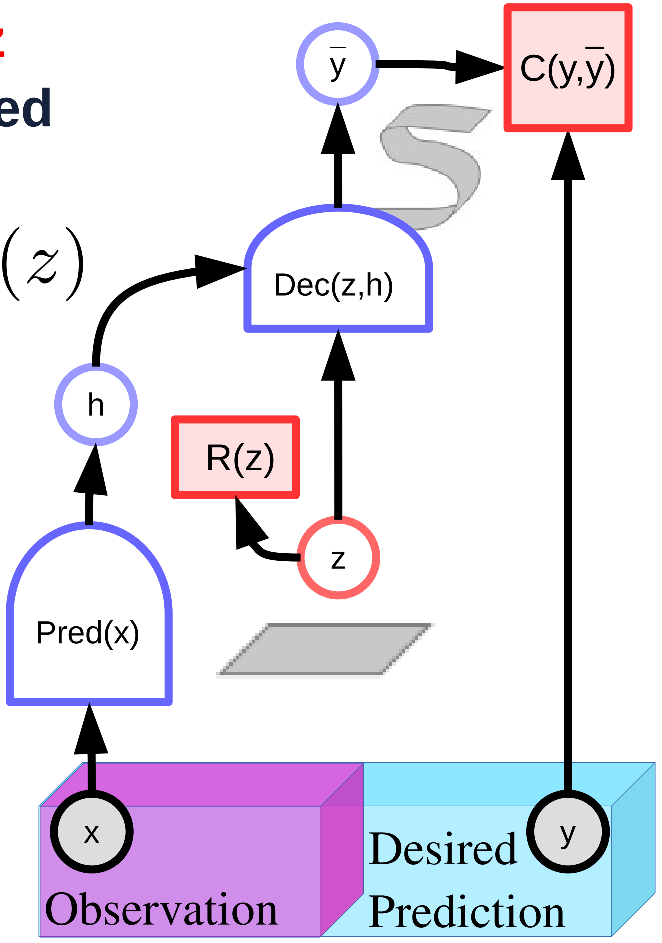


Conditional Latent-Variable EBM

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

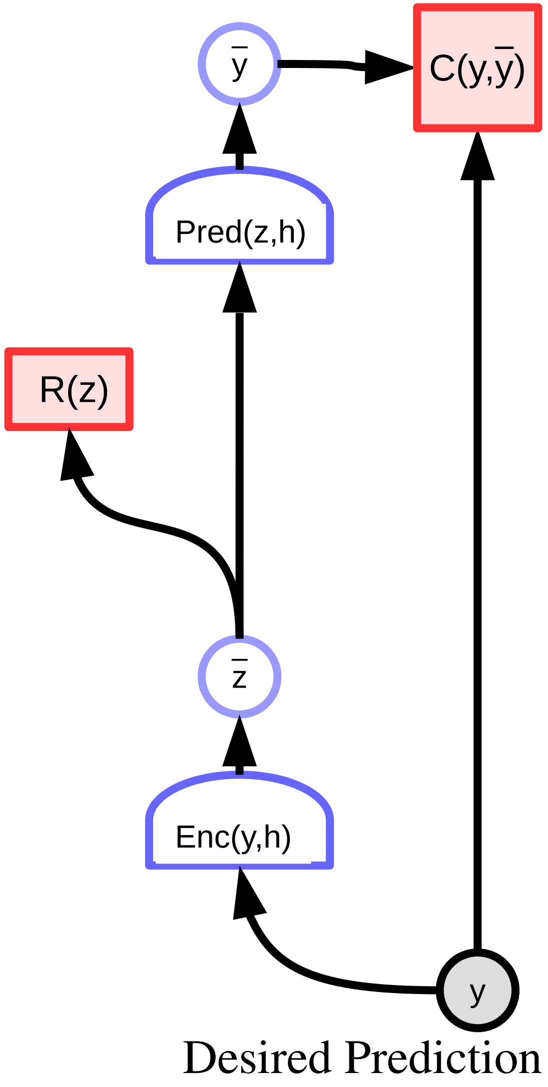
$$E(x,y,z) = C(y, \mathrm{Dec}(\mathrm{Pred}(x), z)) + \lambda R(z)$$

- **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>



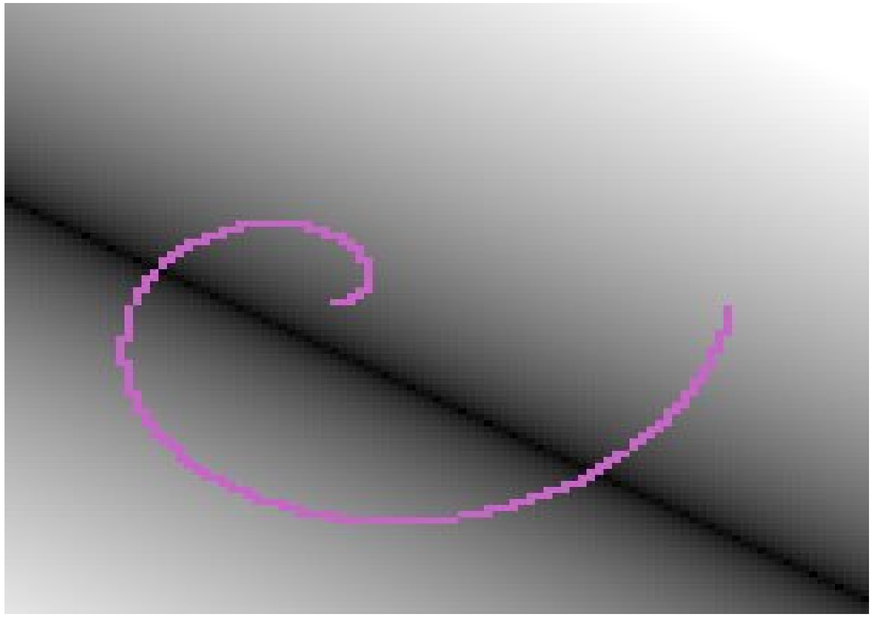
Regularized Auto-Encoders

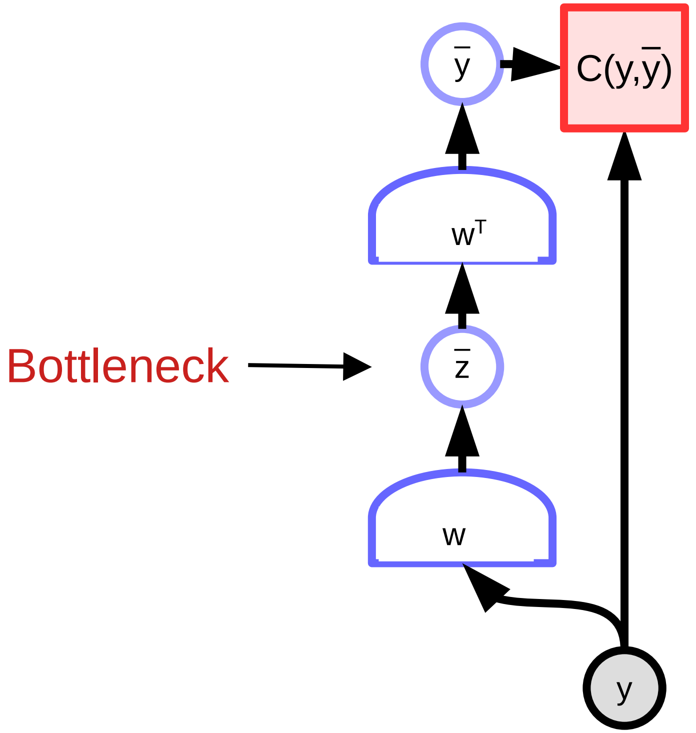
- **Unconditional Models**
- **Regularized Auto-Encoders**
  - Representation computed by encoder
  - No explicit latent variable
- **R(z) now becomes a term in the training loss.**
  - R(z) has no effect on inference
- **Examples:**
  - Bottleneck AE (aka “diabolo” networks)
  - Contracting AE
  - Saturating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



Principal Component Analysis

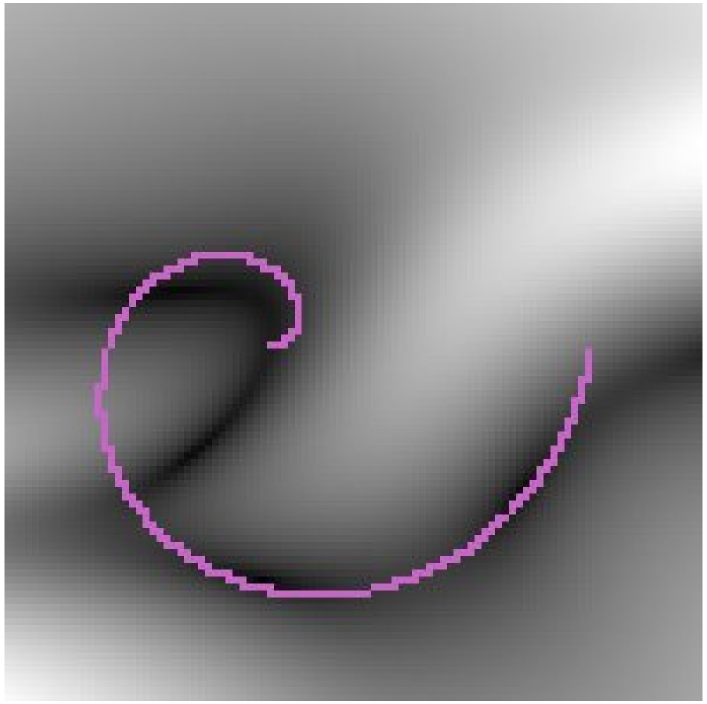
- **PCA is a 2-layer linear auto-encoder with a bottleneck.**
  - Energy: $F_w(y) = ||y - Dec(Enc(y))||^2 = ||y - w^T w||^2$
  - Loss $L(y, w) = F_w(y)$

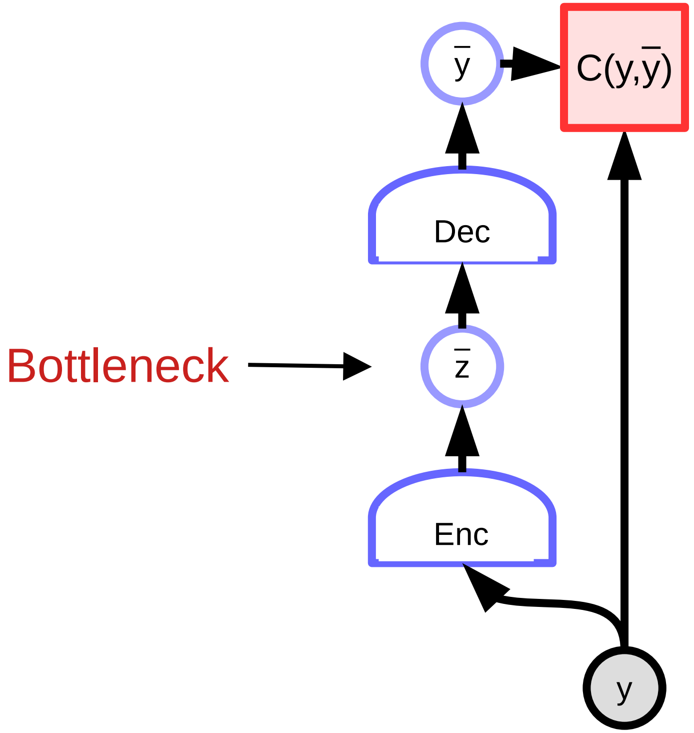




Auto-Encoder with Bottleneck

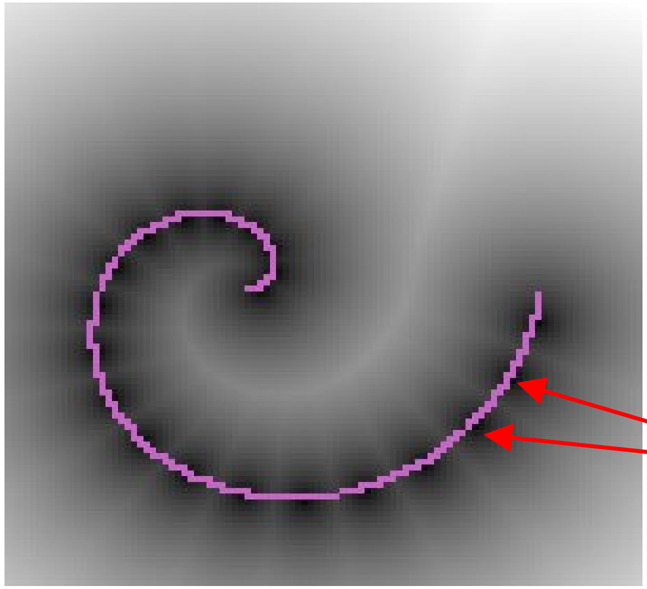
- **non-linear auto-encoder with a bottleneck.**
  - Energy: $F_w(y) = ||y - Dec(Enc(y))||^2$
  - Loss: $L(y, w) = F_w(y)$



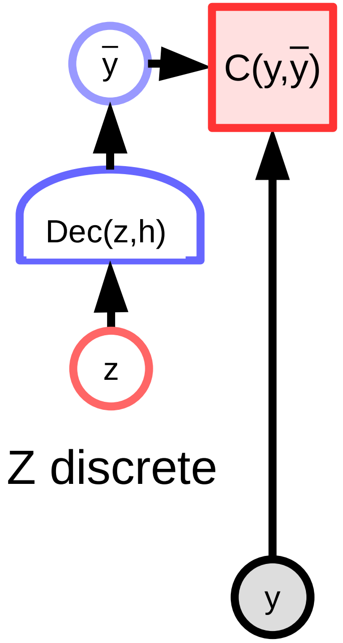


K-Means

- **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: $L(y,w) = F_w(y)$

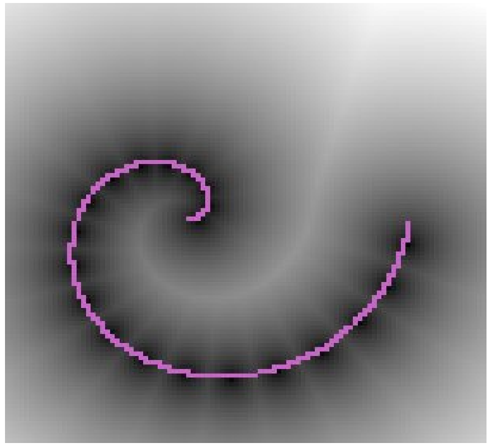


- Latent vector z is constrained to be a 1-hot vector: [0,0,..,01,0,..,0]
- 1 component selects a column of w
- F(y)=0 iff y is equal to a column of w.

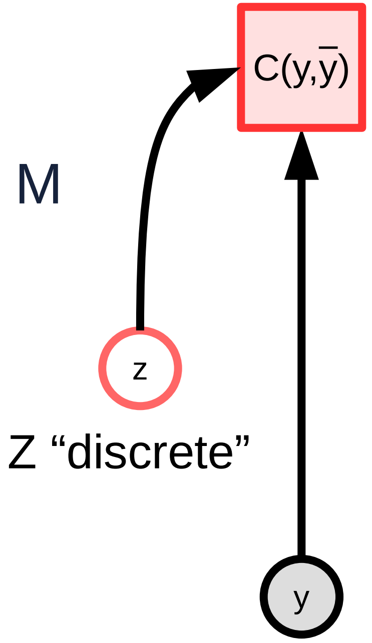


Gaussian Mixture Model

- **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”

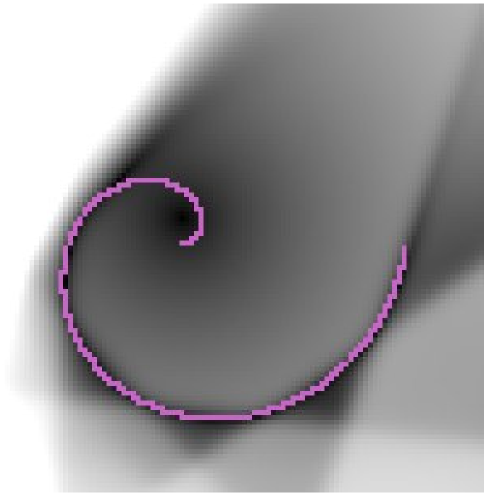


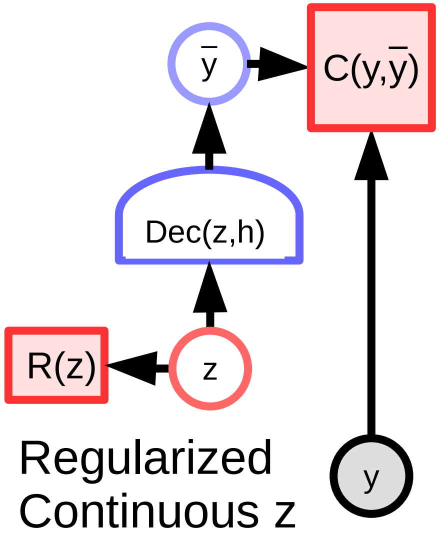
Regularized Latent Variable: Sparse Coding

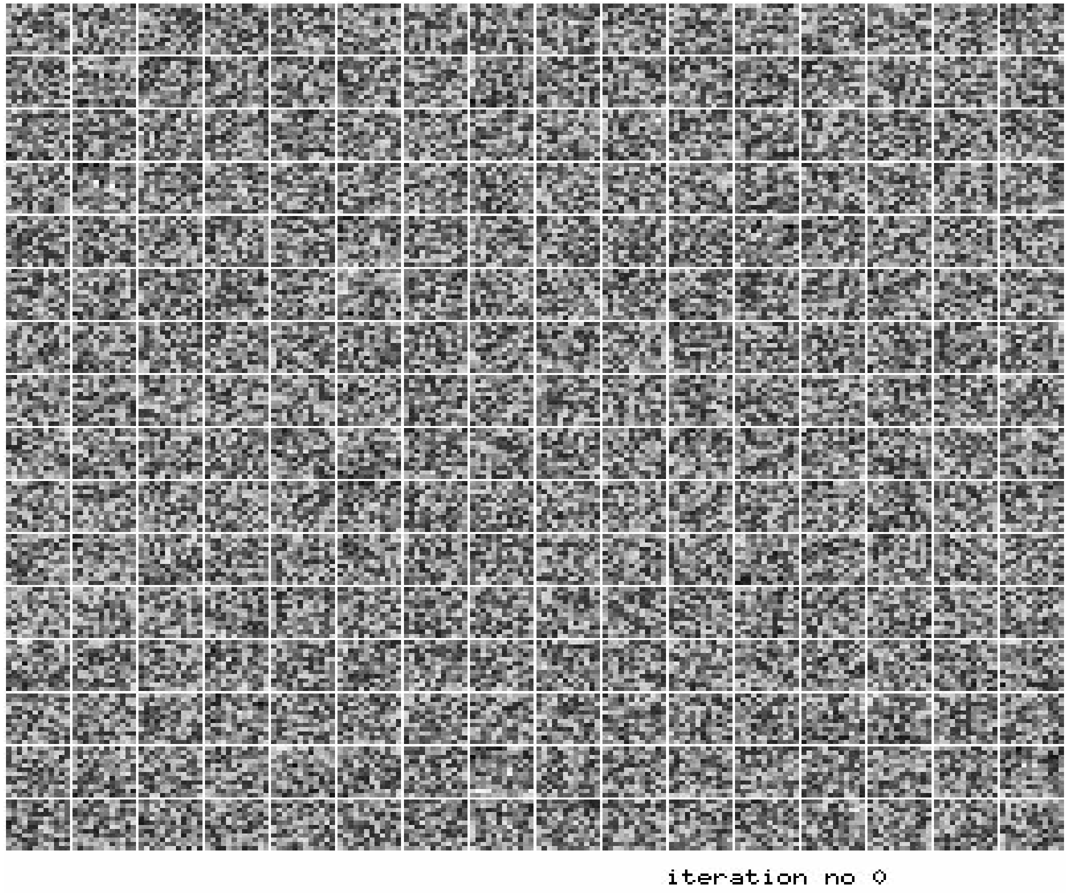
- **A2: regularize the volume of the low energy regions**

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

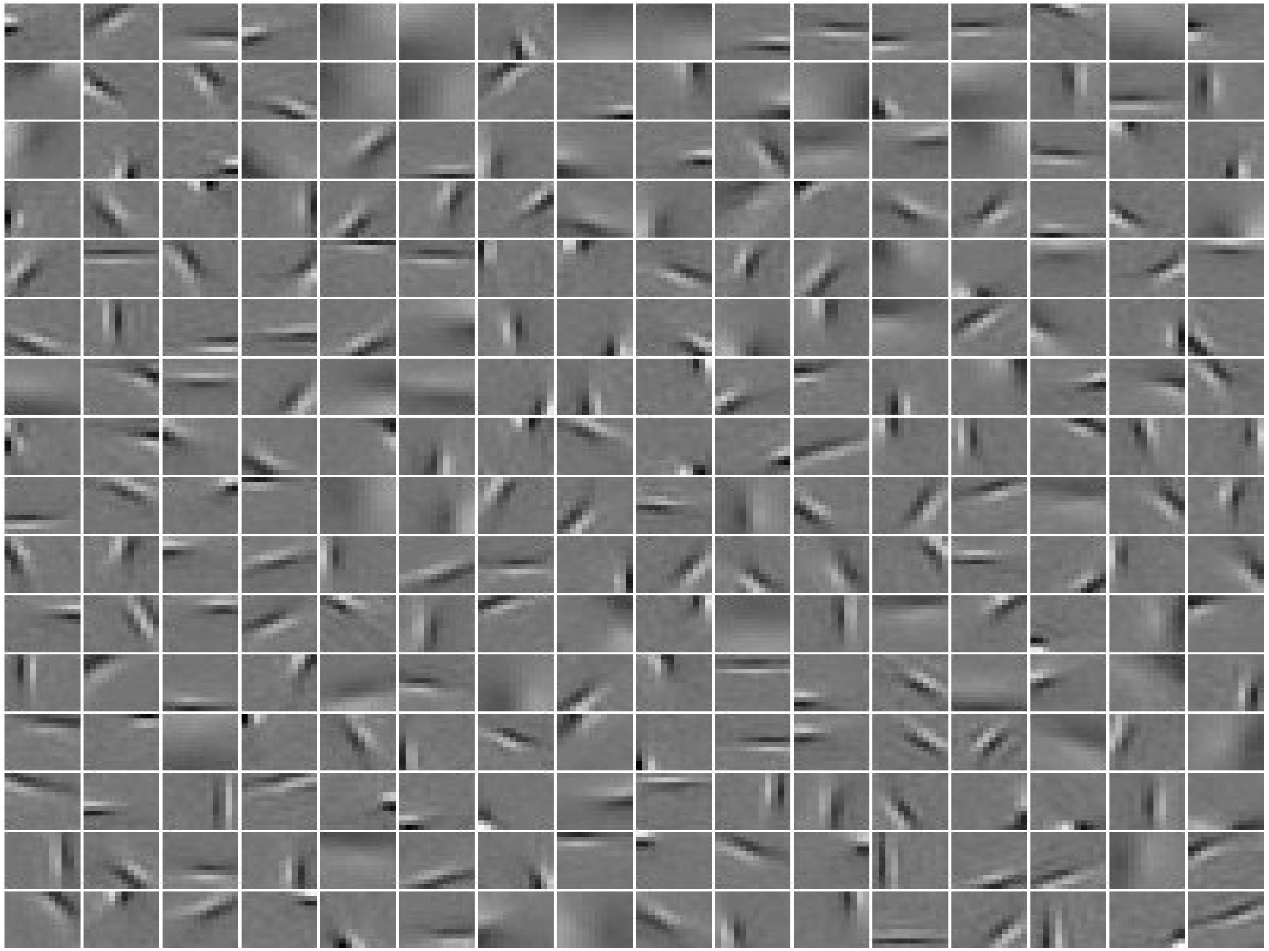
$$F(y) = \min_z E(y,z)$$





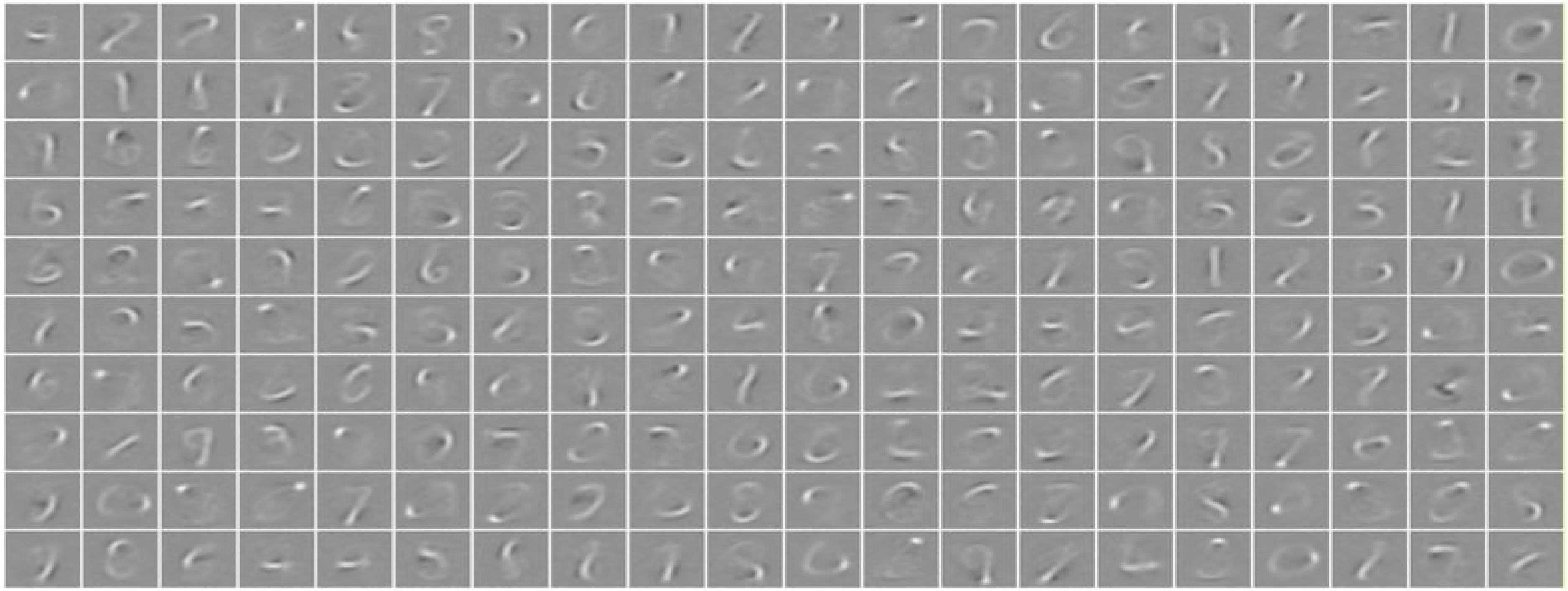


Learned Features on natural patches: V1-like receptive fields



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

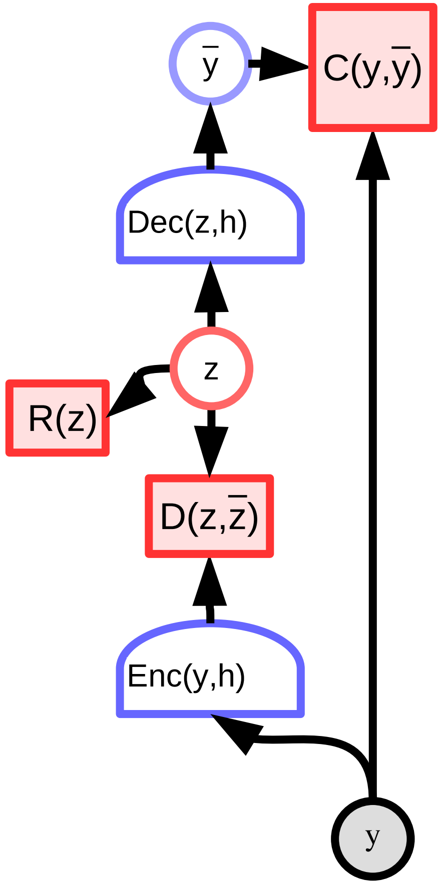
- **Training an encoder to give an approximate solution to the inference optimization problem**

- **Regularized Auto-Encoder, Sparse AE, LISTA**

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

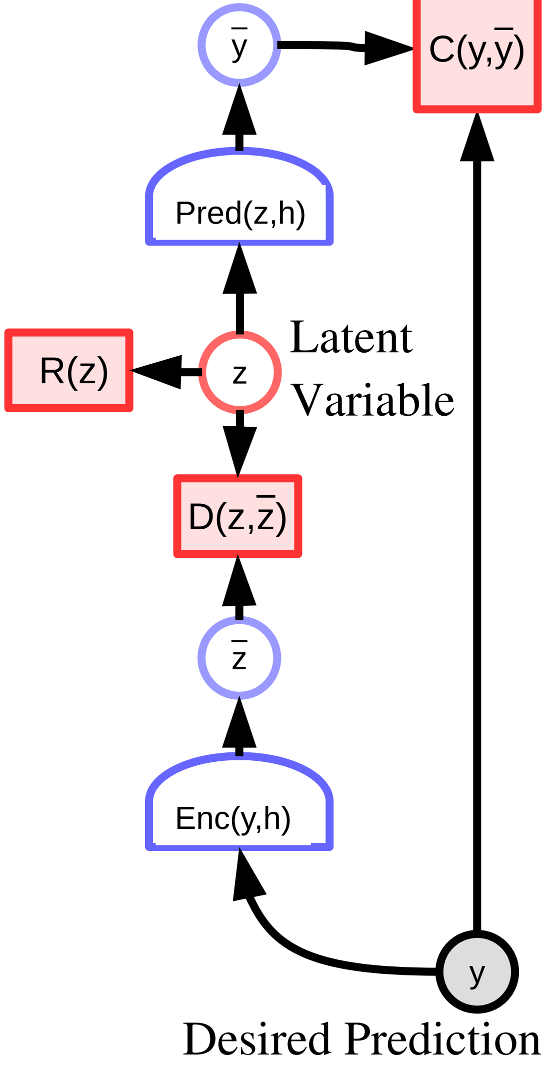
$$F(y) = \min_z E(y, z)$$

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



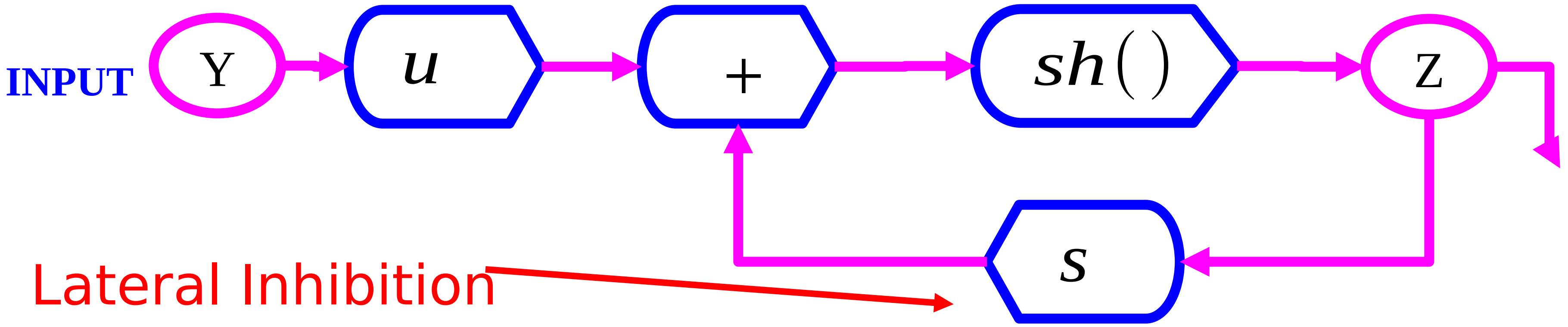
Unconditional LVGEBM with Amortized Inference

- **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**



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

- **ISTA/FastISTA reparameterized:**

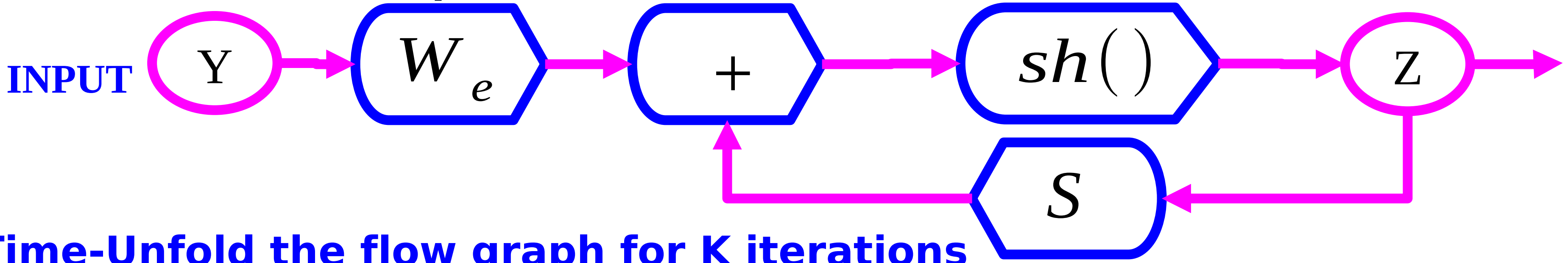
$$z(t+1) = \mathrm{Shrink}_{\alpha\eta}\left[sz(t) + uy\right]; \quad u = \eta w; \quad s = I - \eta w^T w$$

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

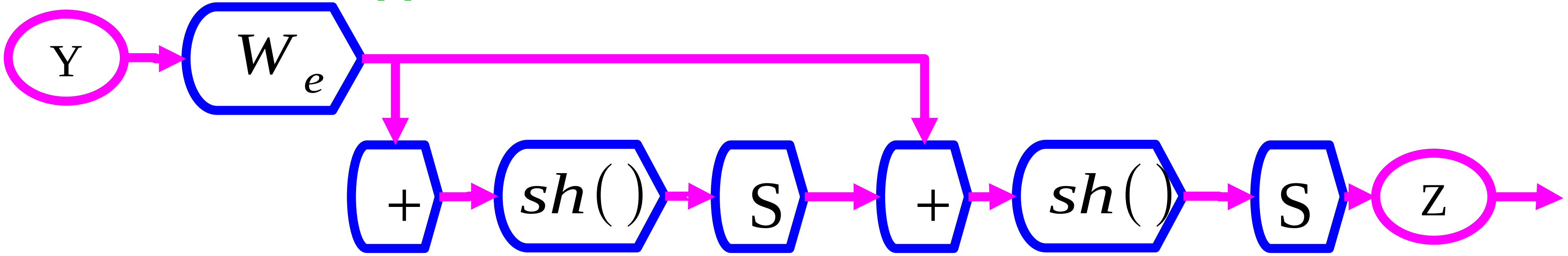
[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]

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**

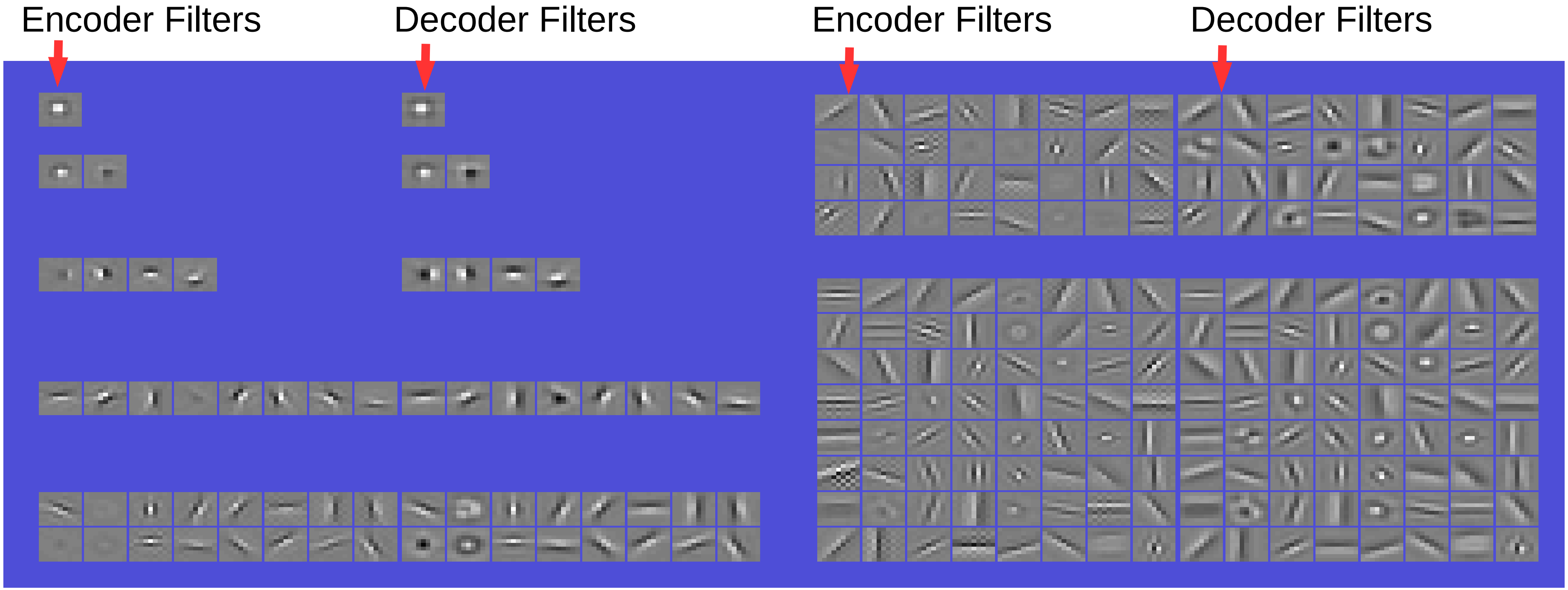


- **Time-Unfold the flow graph for K iterations**
- **Learn the We and S matrices with “backprop-through-time”**
- **Get the best approximate solution within K iterations**



Convolutional Sparse Auto-Encoder on Natural Images

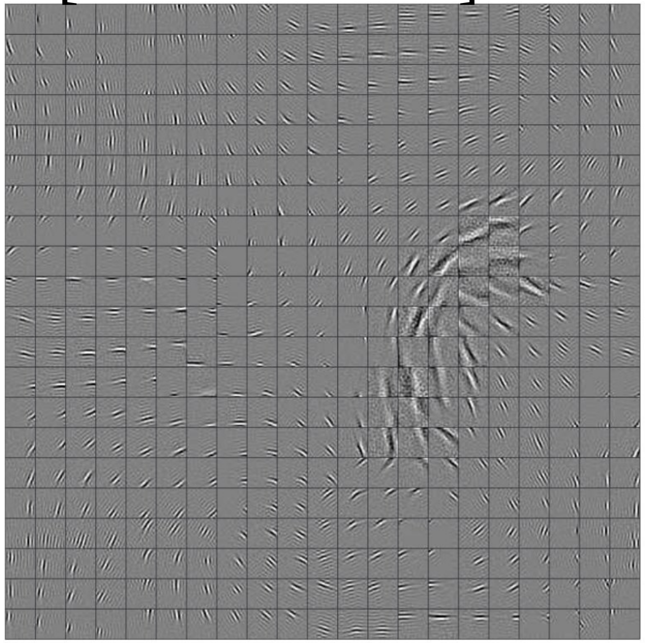
- **Filters and Basis Functions obtained. Linear decoder (conv)**
  - with 1, 2, 4, 8, 16, 32, and 64 filters [Kavukcuoglu NIPS 2010]



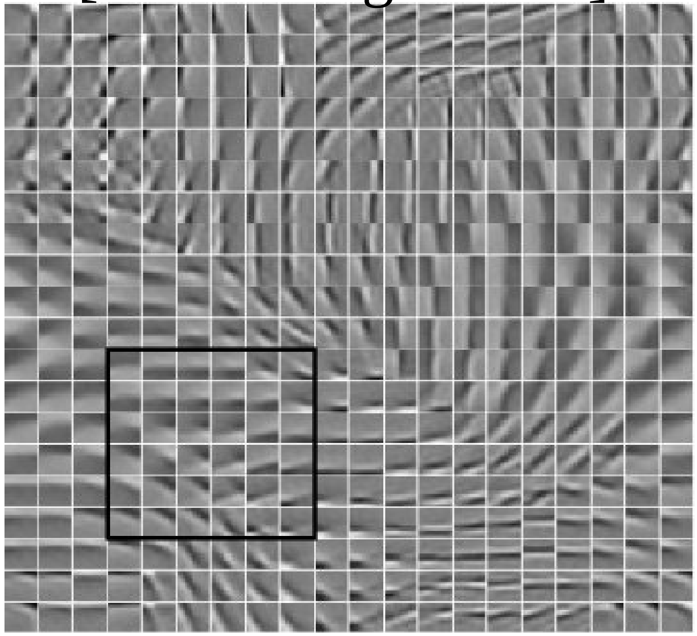
Learning invariant features

- **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].

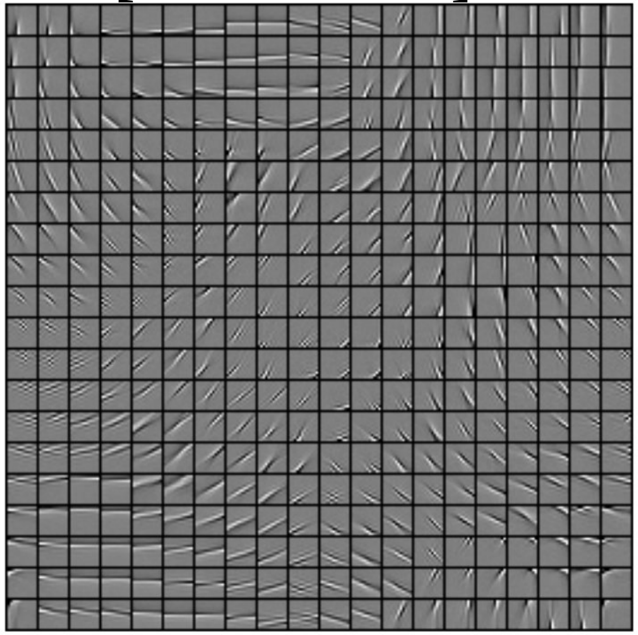
[Osindero 2006]



[Kavukcuoglu 2009]

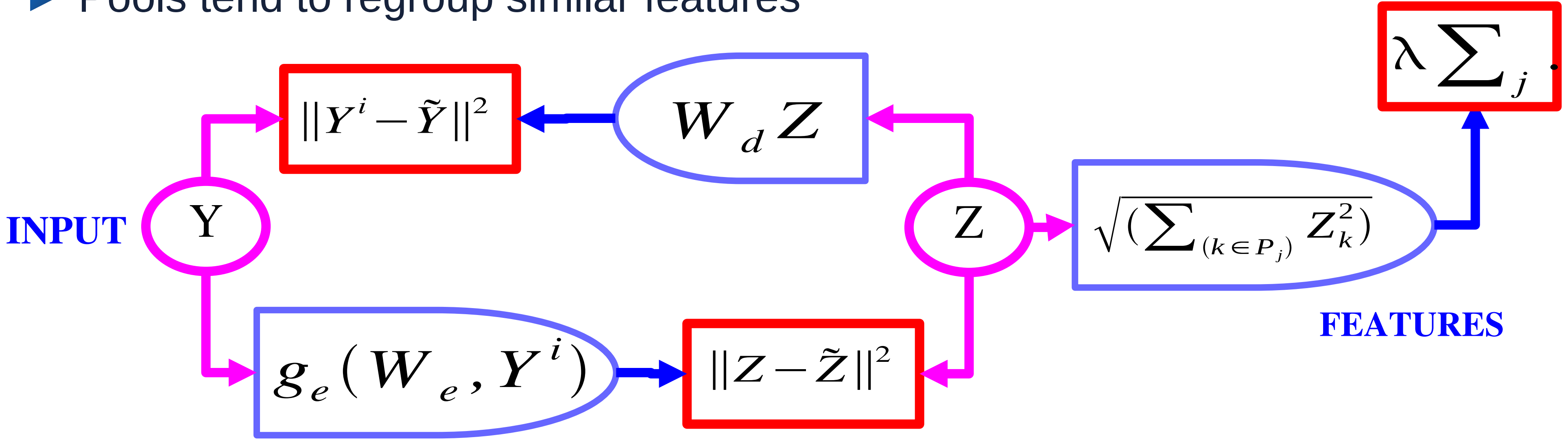


[Mairal 2011]



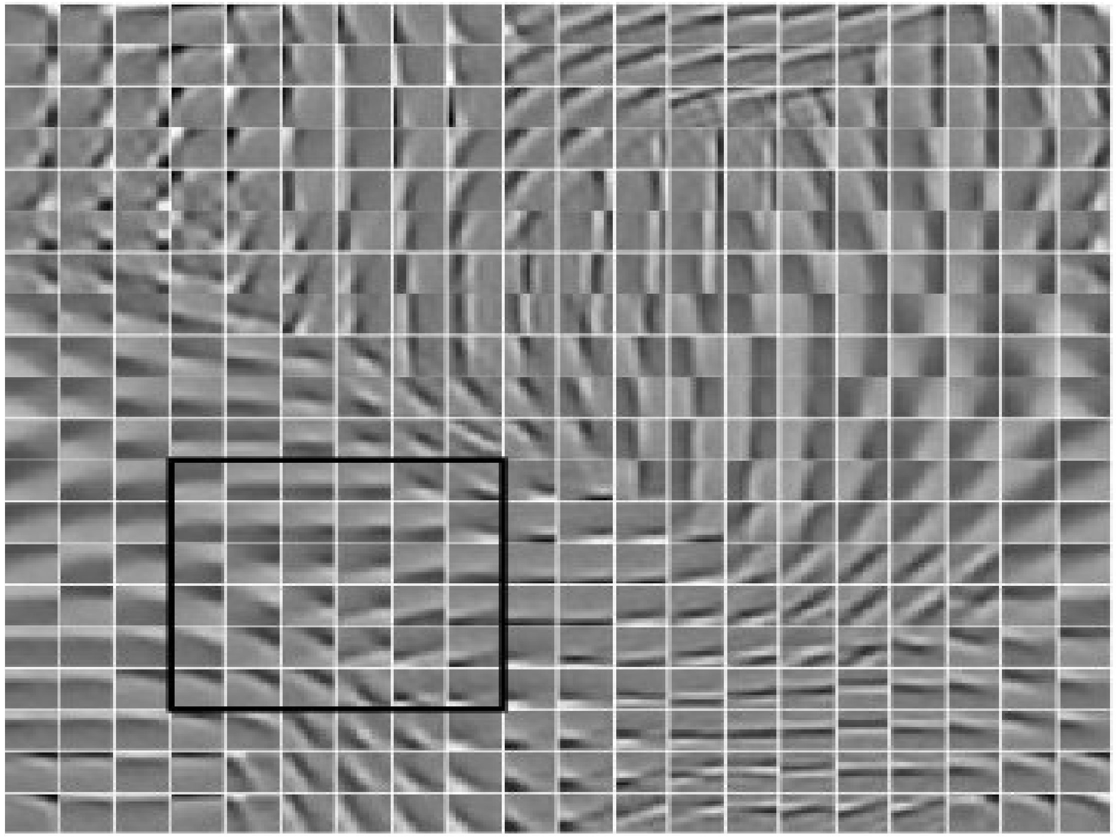
AE with Group Sparsity [Kavukcuoglu et al. CVPR 2009]

- **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



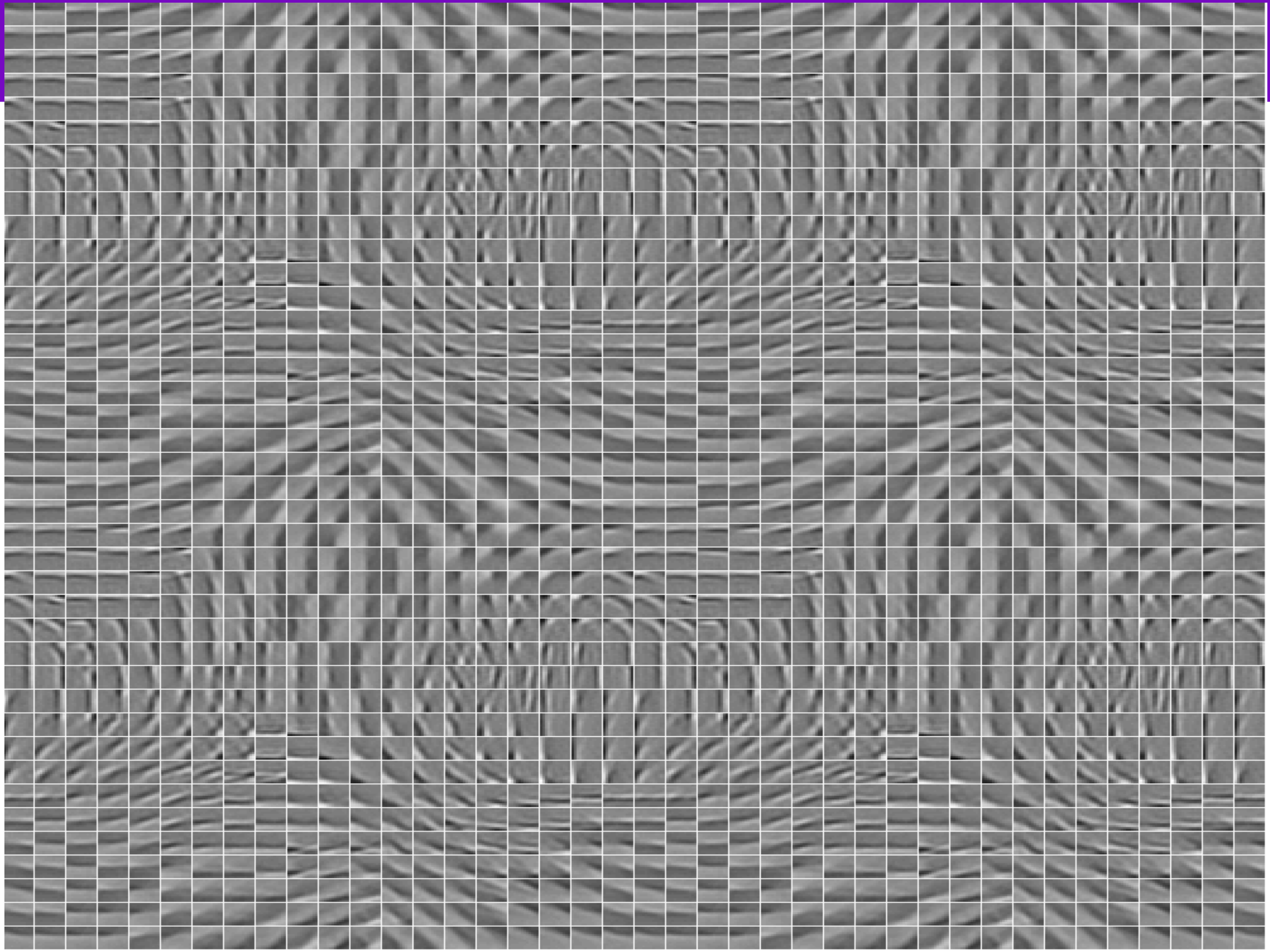
Pooling over features in a topographic map.

- **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.



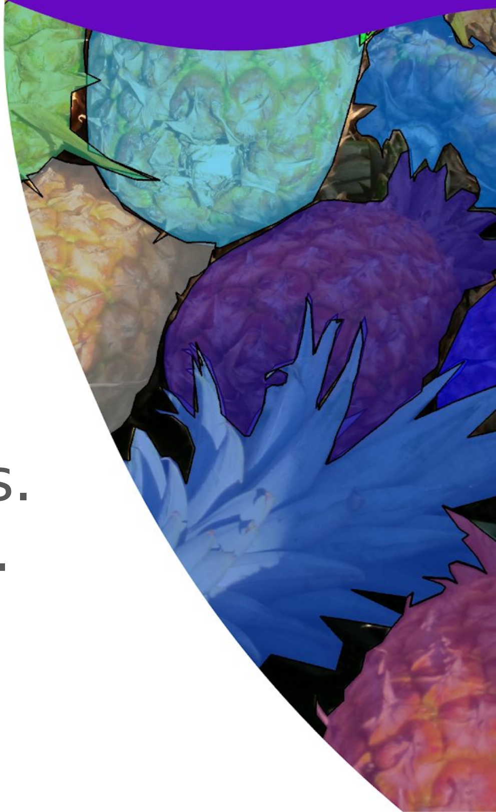
Pinwheels!

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



Regularization through (variational) marginalization.

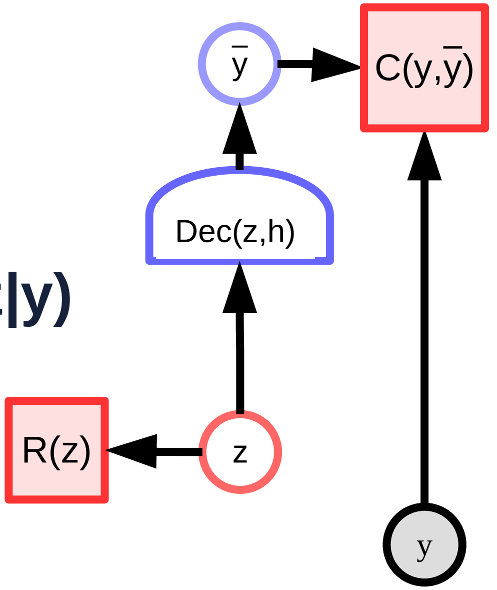
Push down on the energy of training samples.
Minimize the capacity of the latent variables.
Maximize the capacity of the representation.



Making z a noisy variable to reduce its information content

- **The information content of the latent variable z must be minimized**
- **One (probabilistic) way to do this:**
  - make z “fuzzy” (e.g. stochastic)
  - Z is a sample from a distribution q(z|y)

$$E(y,z) = C(y, Dec(z))$$



- **Minimize the expected value of the energy under q(z|y)**

$$< E(y) >= \int_z q(z|y) E(y,z)$$

- **Minimize the information content of q(z|y) about y**

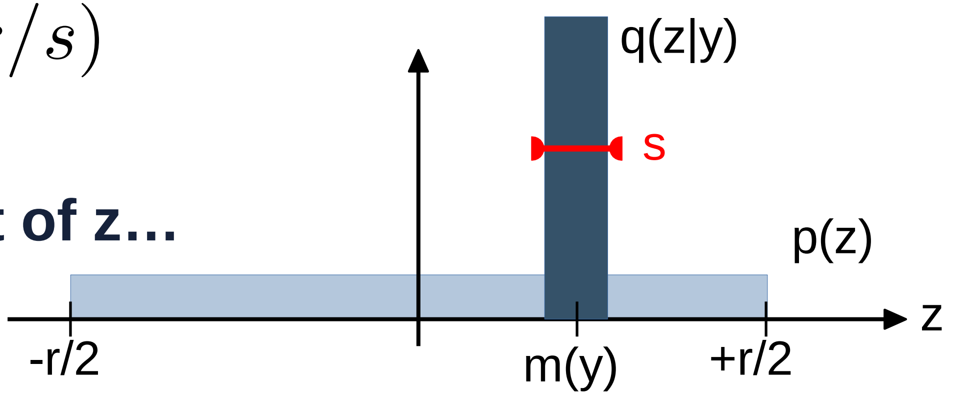
What information does q(z|y) give us about y?

- **Suppose that all the z come from a distribution p(z)**
  - e.g. p(z) uniform over a hypercube of dimension d: $[-r/2, +r/2]^d$
- **Suppose that q(z|y) is uniform**
  - over a small hypercube of size s centered on m(y)
  - e.g. q(z|y) uniform over $[m_i(y)-s/2, m_i(y)+s/2]$ in each dimension i.
- **There are $(r/s)^d$ small cubes in the big cube**
  - Hence each small cube gives

$$H(z|y) = \log_2(r/s)^d = d\log_2(r/s)$$

bits of information about y.

- **To minimize the information content of z…**
  - I can make the small cube large
  - I can make the large cube small



What information does q(z|y) give us about y?

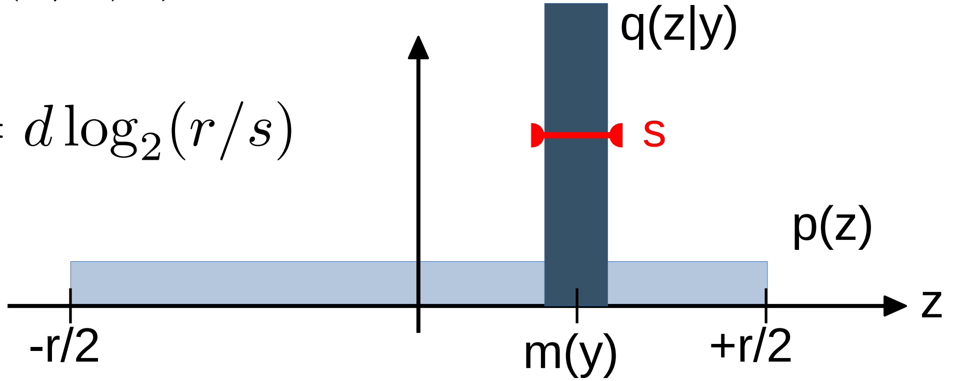
- Suppose that all the z come from a distribution p(z)
- Suppose that each z distributes according to q(z|y)
- The amount of information that q(z|y) gives about p(z) is

$$KL(q(z|y), p(z)) = \int_z q(z|y) \log_2(q(z|y)/p(z))$$

- Example: uniform case: p(z) = $(1/r)^d$, q(z|y) = $(1/s)^d$

$$KL(q(z|y), p(z)) = \int_z q(z|y) \log_2((1/s)^d/(1/r)^d)$$

$$KL(q(z|y), p(z)) = \log_2(r/s)^d \int_z q(z|y) = d \log_2(r/s)$$



General case: minimize expected energy & information of z on y

- **Minimize the expected energy**

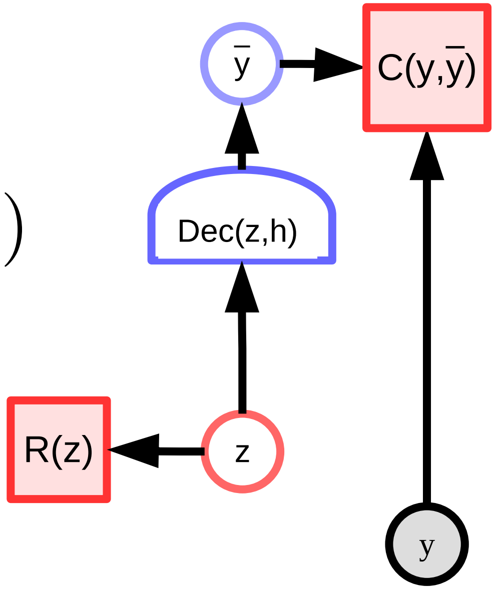
$$< E(y) >_q = \int_z q(z|y)E(y,z)$$

$$E(y,z) = C(y, Dec(z))$$

- **Minimize the relative entropy**
  - Between q(z|y) and a prior distribution p(z).

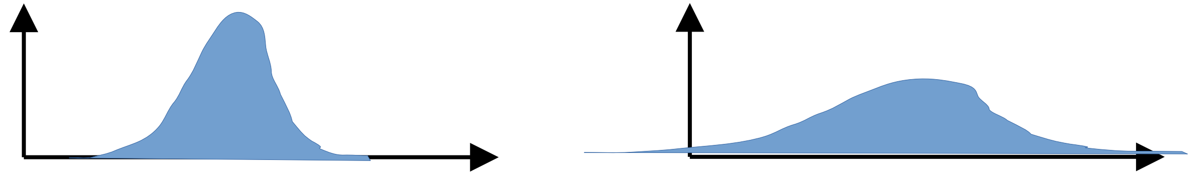
$$KL(q(z|y), p(z)) = \int_z q(z|y) \log_2(q(z|y)/p(z))$$

  - This is the number of bits one sample from q(z|y) will give us about p(z)



Marginalization as Regularization through Maximum Entropy

- **Find a distribution q(z|y) that minimizes the expected energy while having maximum entropy**
  - high entropy distribution == small information content from a sample



  - Pick a family of distributions q(z|y) (e.g. Gaussians) and find the one that minimizes the **variational free energy**:

$$\tilde{F}_q(y) = \int_z q(z|y)E(y,z) + \frac{1}{\beta}\int_z q(z|y)\log_2(q(z|y)/p(z))$$

  - The trade-off between energy and entropy is controlled by the beta parameter.

Gaussian case

- **Both p(z) and q(z|y) are Gaussians**

$$p(z) = N(0, r) \qquad q(z|y) = N(m(y), s(y))$$

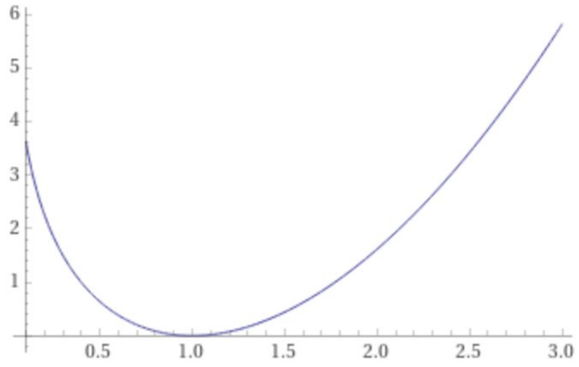
$$KL(q,p) = \log(r/s) + \frac{s^2 + m(y)^2}{2r^2} - \frac{1}{2}$$

(this is in nats, not bits. Divide by log(2) to get it in bits).

- **Assume r=1:**

$$KL(q,p) = \frac{1}{2}(s^2 + m(y)^2 - \log_2(s^2) - 1)$$

  - This has a minimum at s=1



Variational Auto-Encoder

- **If Q(y,z) is quadratic, q(z|y) is Gaussian.**

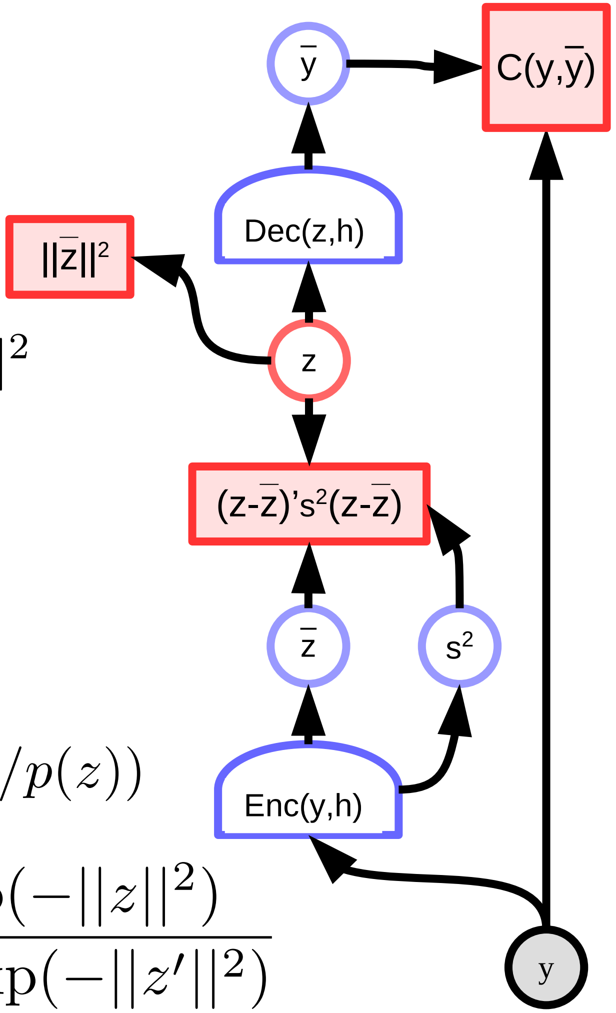
$$Q_{w_e}(y,z) = (z - \mathrm{Enc}_{w_e}(y))^T s^2 (z - \mathrm{Enc}_{w_e}(y)) + \gamma ||\mathrm{Enc}_{w_e}(y)||^2$$

$$q_{w_e}(z|y) = \frac{e^{-\beta Q_{w_e}(y,z)}}{\int_{z'} e^{-\beta Q_{w_e}(y,z')}}$$ **(Gaussian)**

$$\tilde{F}_w(y) = \int_z q_{w_e}(z|y) E_{w_d}(y,z) + \frac{1}{\beta} \int_z q_{w_e}(z|y) \log(q_{w_e}(z|y)/p(z))$$

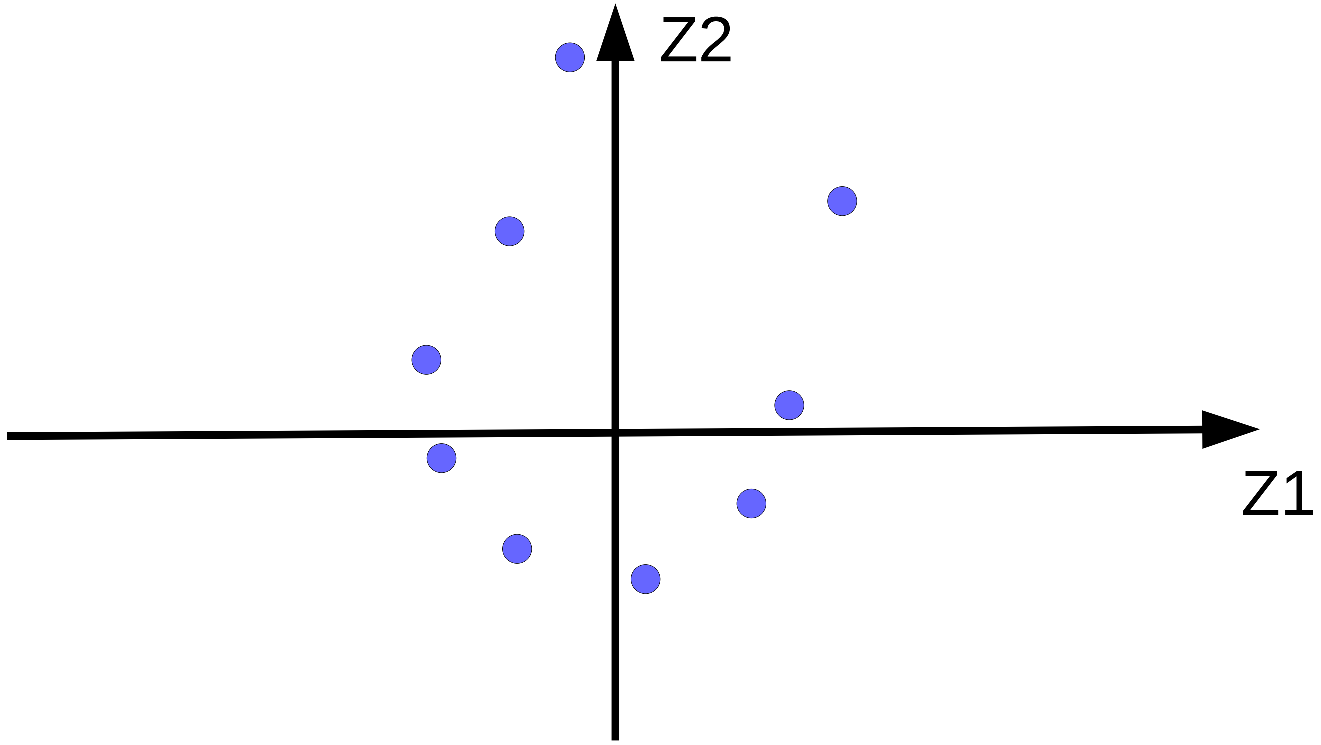
- **Loss** $\mathcal{L}(y,w) = \tilde{F}_w(y)$

$$p(z) = \frac{\exp(-||z||^2)}{\int_{z'} \exp(-||z'||^2)}$$



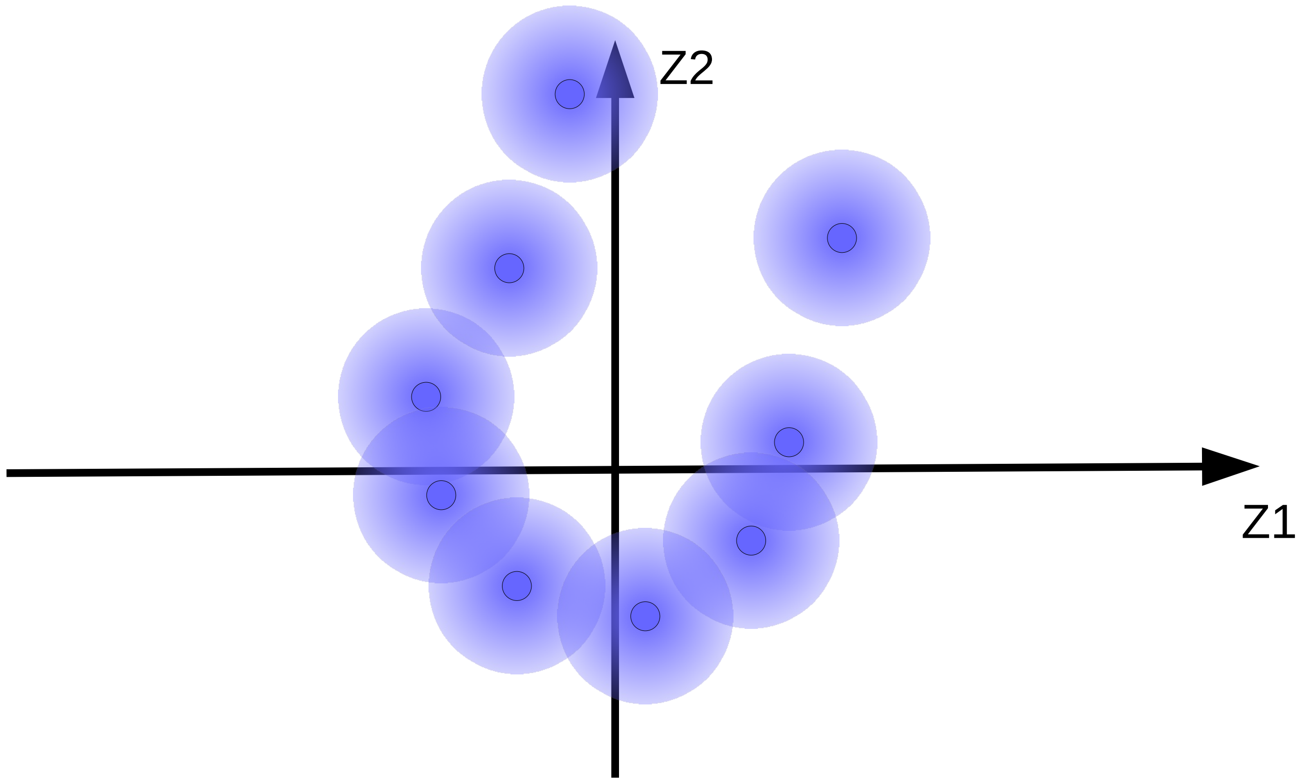
Variational Auto-Encoder

- **Code vectors for training samples**



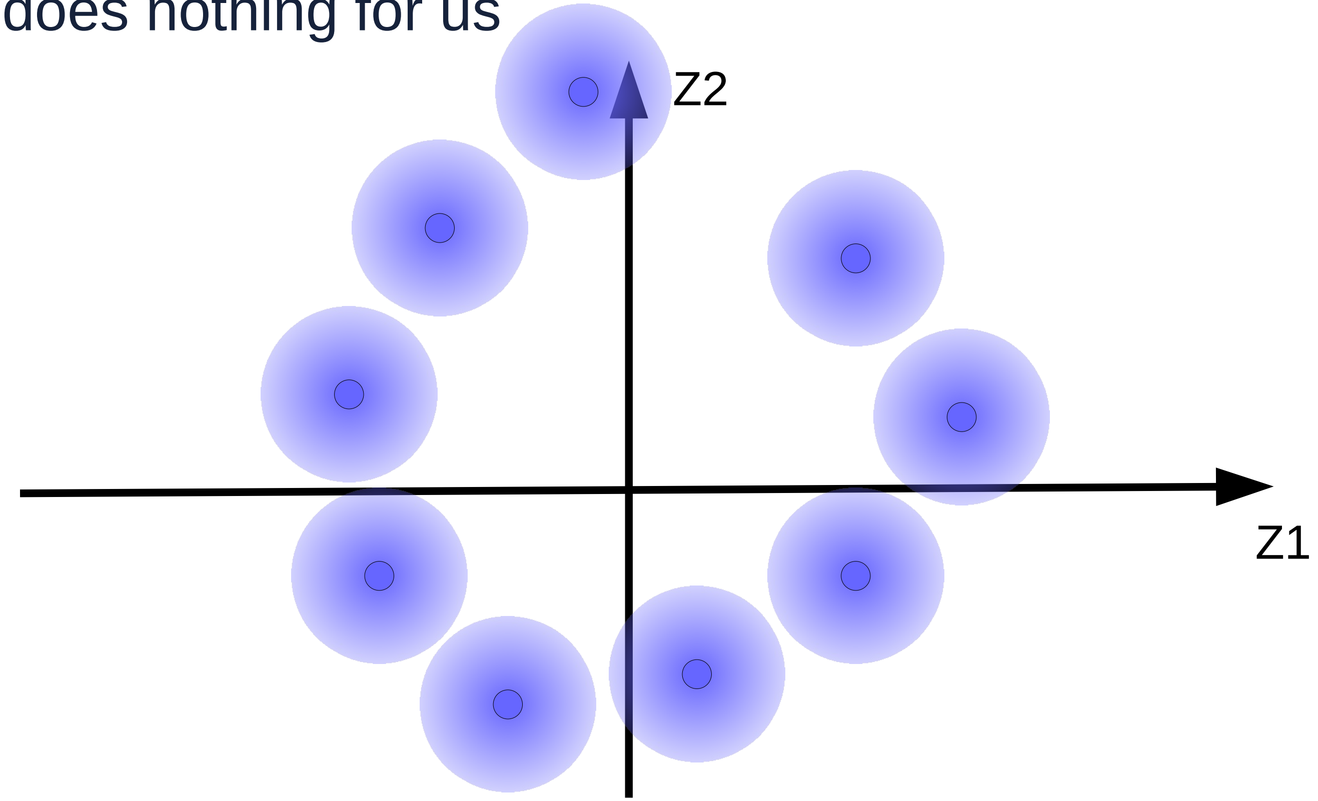
Variational Auto-Encoder

- **Code vectors for training sample with Gaussian noise**
  - Some fuzzy balls overlap, causing bad reconstructions



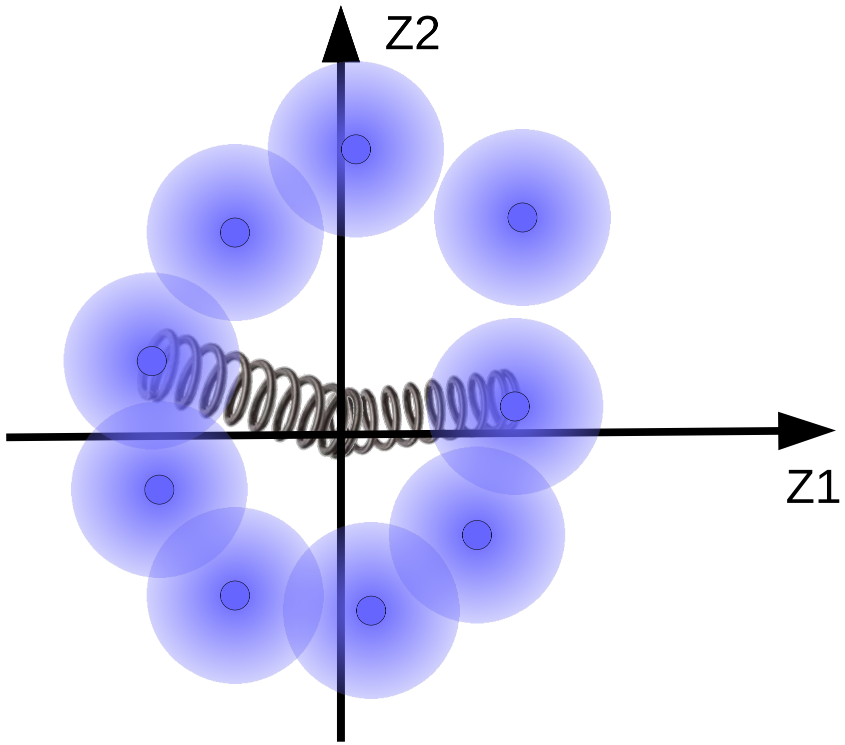
Variational Auto-Encoder

- **The code vectors want to move away from each other to minimize reconstruction error**
  - But that does nothing for us



Variational Auto-Encoder

- **Attach the balls to the center with a sping, so they don’t fly away**
  - Minimize the square distances of the balls to the origin
- **Center the balls around the origin**
  - Make the center of mass zero
- **Make the sizes of the balls close to 1 in each dimension**
  - Through a so-called KL term

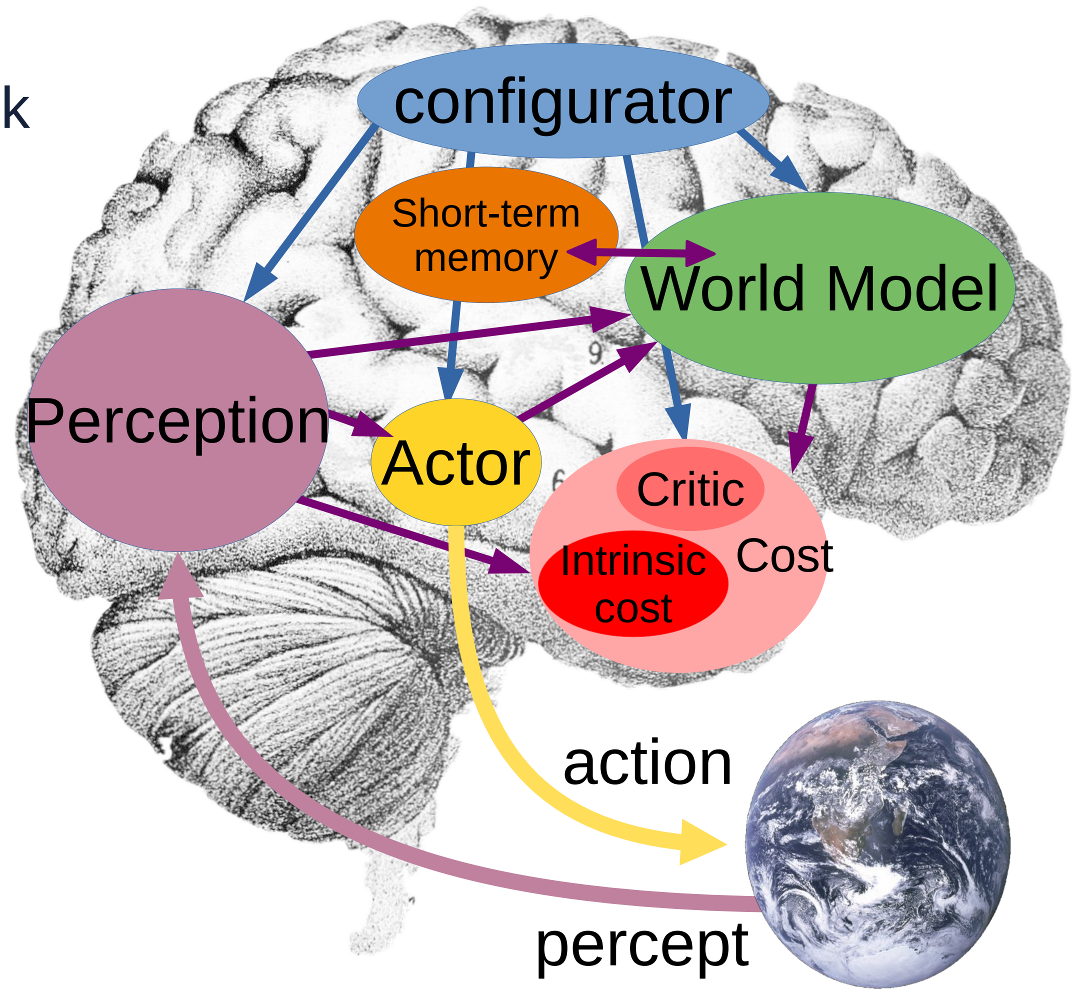


FACEBOOK AI

Architecture of Autonomous AI

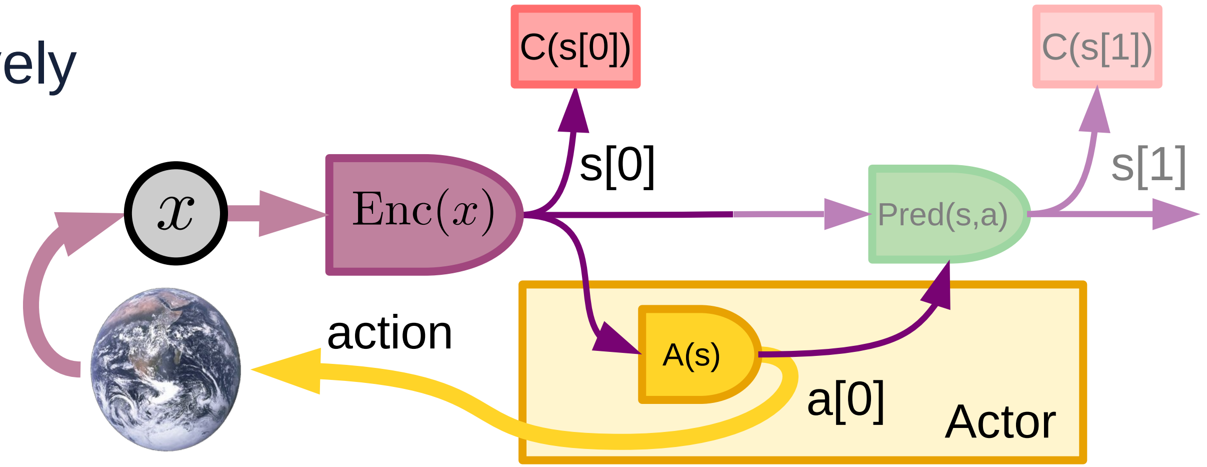
Modular Architecture for Autonomous AI

- **Configurator**
  - Configures other modules for task
- **Perception**
  - Estimates state of the world
- **World Model**
  - Predicts future world states
- **Cost**
  - Compute "discomfort"
- **Actor**
  - Find optimal action sequences
- **Short-Term Memory**
  - Stores state-cost episodes



Mode-1 Perception-Action Cycle

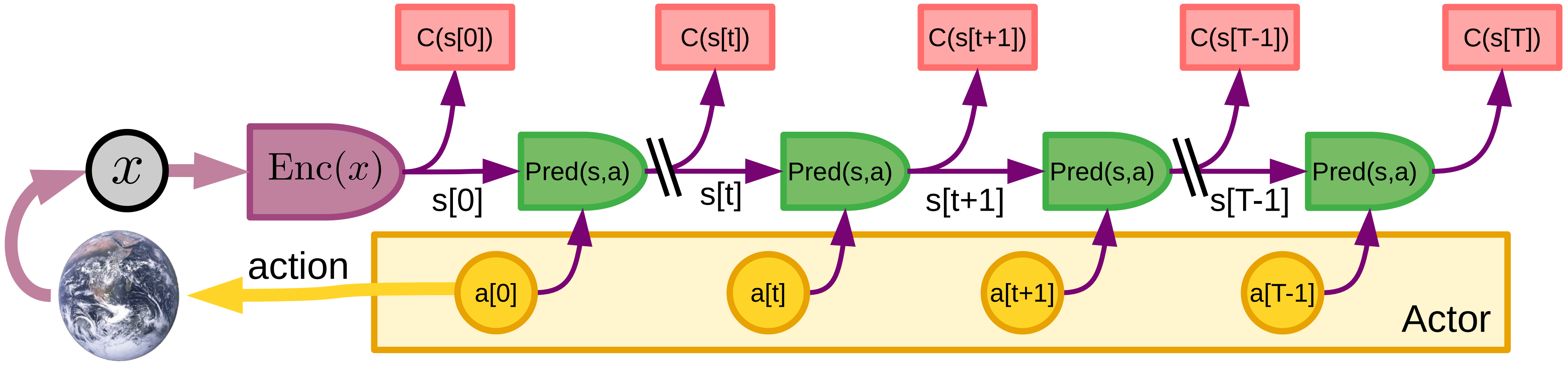
- **Perception module s[0]=Enc(x)**
  - Extract representation of the world
- **Policy module A(s[0])**
  - Computes an action reactively
- **Cost module C(s[0])**
  - Computes cost of state
- **Optionally:**
  - World Model Pred(s,a)
  - Predicts future state
  - Stores states and costs in short-term memory



Y. LeCun

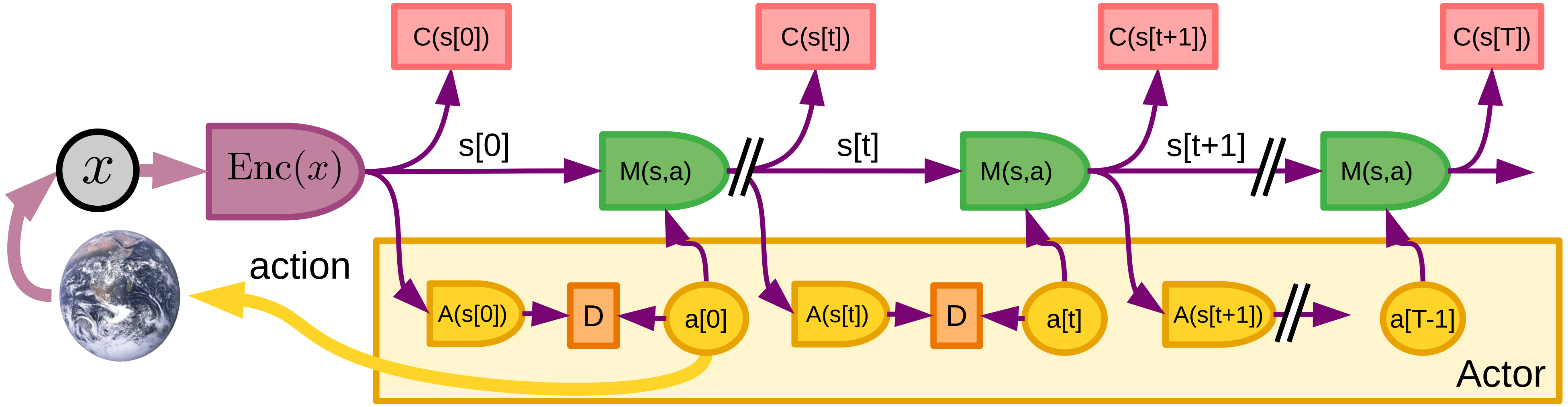
Mode-2 Perception-Planning-Action Cycle

- **Akin to Model-Predictive Control (MPC)**
- **Actor proposes an ation sequence**
- **World Model predicts outcome**
- **Actor optimizes action sequence to minimize cost**
  - e.g. using gradient descent, dynamic programming, MC tree search…
- **Actor sends first action(s) to effectors**



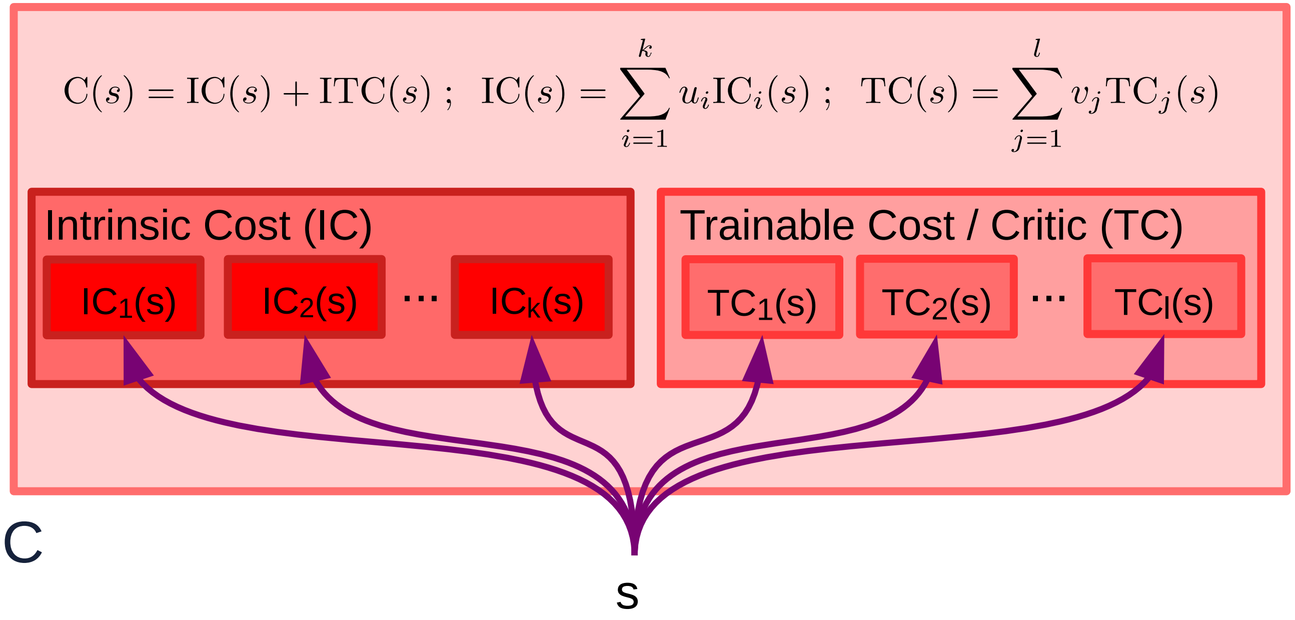
Compiling Mode-2 into Mode-1

- Akin to Amortized Inference
- System performs Mode-2 cycle to get optimal action sequence.
- Optimal actions used as targets to train the policy module A(s)
- Policy module can be used for Mode-1 or to initialize Mode-2.



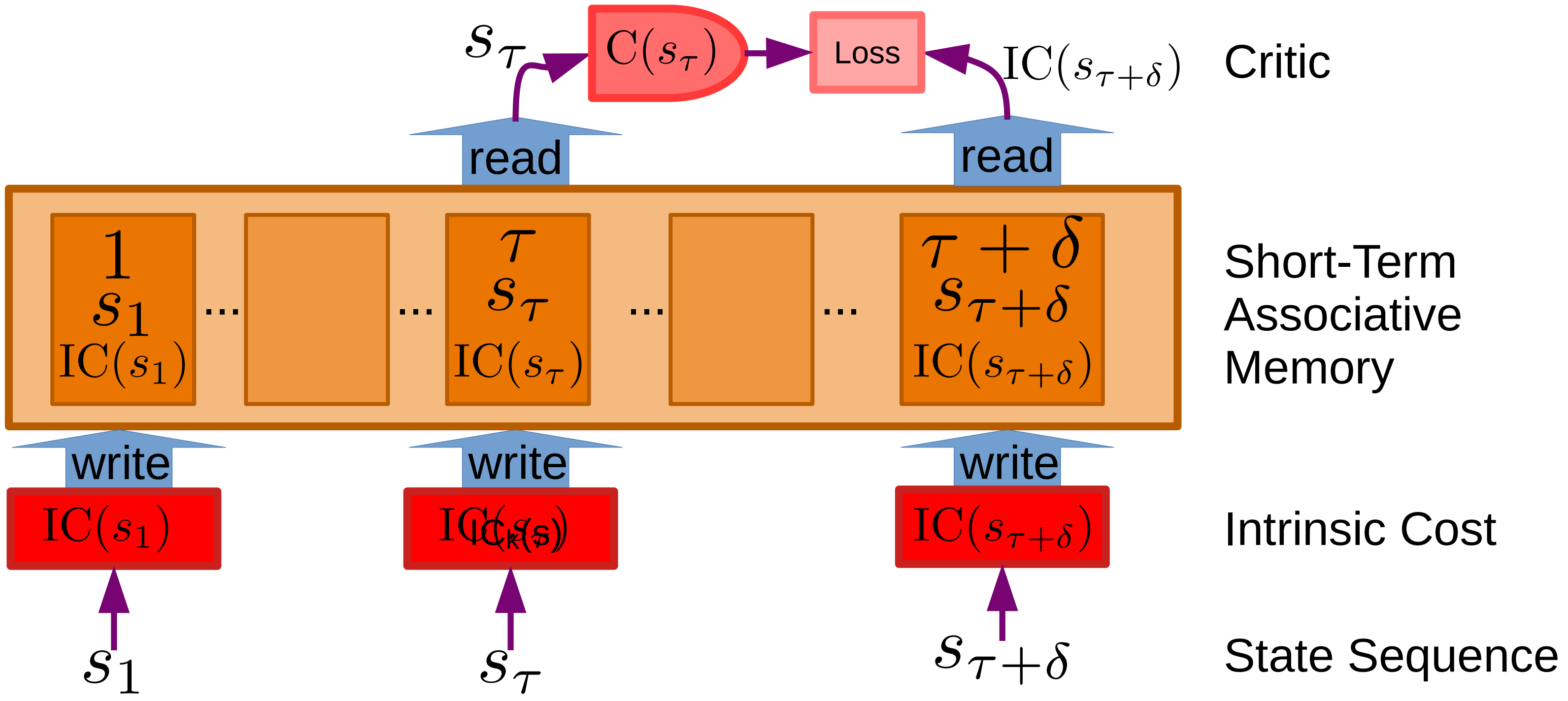
Cost Module

- **Intrinsic Cost (IC)**
  - Immutable cost modules.
  - Hard-wired drives and behaviors
- **Trainable Cost (TC)**
  - Trainable
  - Predicts future values of IC
  - Equivalent to a critic in RL
  - Implements subgoals
  - Configurable
- **All are differentiable**


$C(s) = \mathrm{IC}(s) + \mathrm{ITC}(s)\ ;\ \ \mathrm{IC}(s) = \sum_{i=1}^{k} u_i \mathrm{IC}_i(s)\ ;\ \ \mathrm{TC}(s) = \sum_{j=1}^{l} v_j \mathrm{TC}_j(s)$
Intrinsic Cost (IC)
$\mathrm{IC}_1(s)$
$\mathrm{IC}_2(s)$
...
$\mathrm{IC}_k(s)$
Trainable Cost / Critic (TC)
$\mathrm{TC}_1(s)$
$\mathrm{TC}_2(s)$
...
$\mathrm{TC}_l(s)$
s

Training the Critic

- **Critic is trained to predict future values of the intrinsic cost from the current state**
  - Uses the short term memory to produce training pairs.



s_τ
$\mathrm{C}(s_\tau)$
Loss
$\mathrm{IC}(s_{\tau+\delta})$
Critic
read
read
1 s_1 $\mathrm{IC}(s_1)$
...
τ s_τ $\mathrm{IC}(s_\tau)$
$\tau+\delta$ $s_{\tau+\delta}$ $\mathrm{IC}(s_{\tau+\delta})$
Short-Term Associative Memory
write
write
write
$\mathrm{IC}(s_1)$
$\mathrm{IC}(s_\tau)$
$\mathrm{IC}(s_{\tau+\delta})$
Intrinsic Cost
s_1
s_τ
$s_{\tau+\delta}$
State Sequence

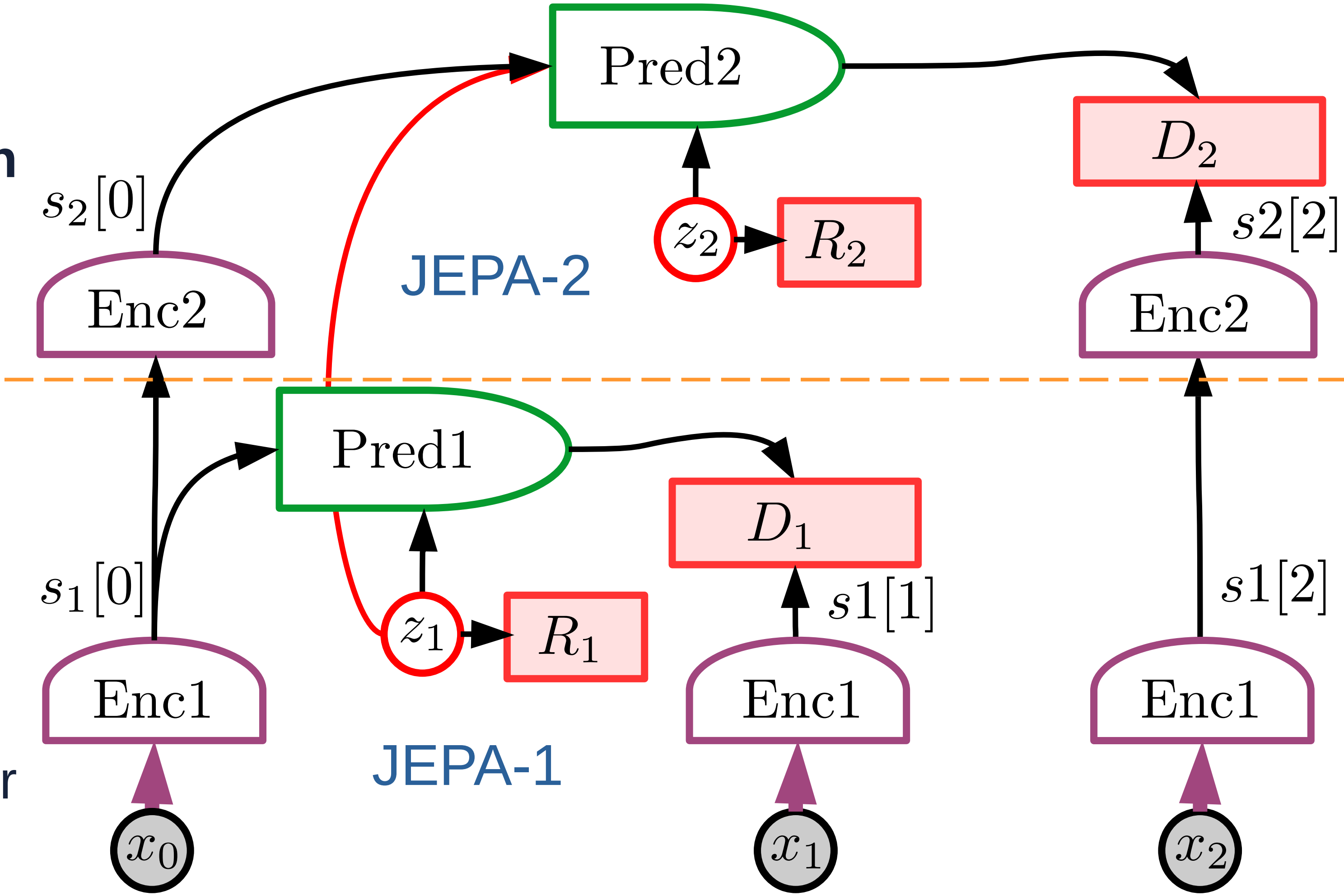
FACEBOOK AI

Hierarchical Planning with H-JEPA

For control, planning, and policy learning.

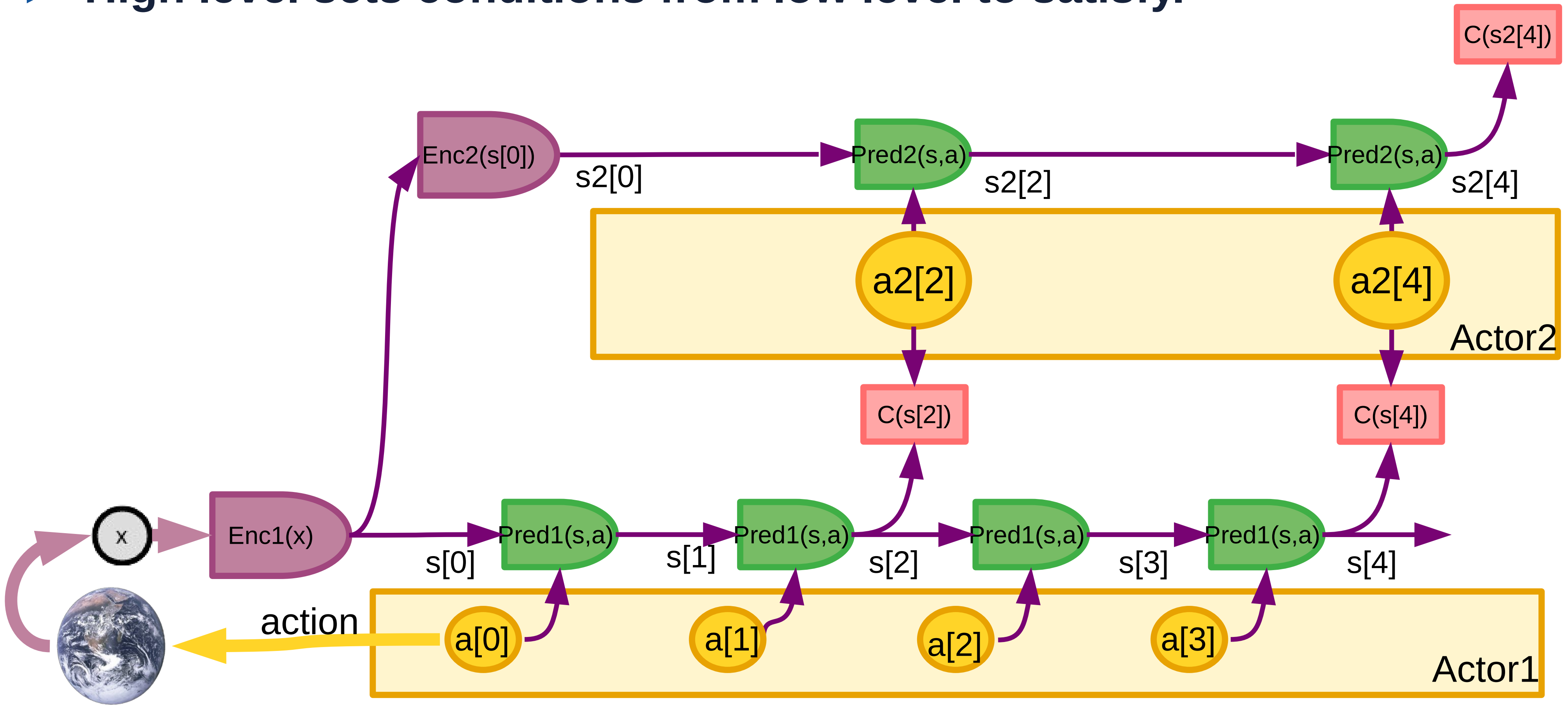
Multi time-scale Predictions

- **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



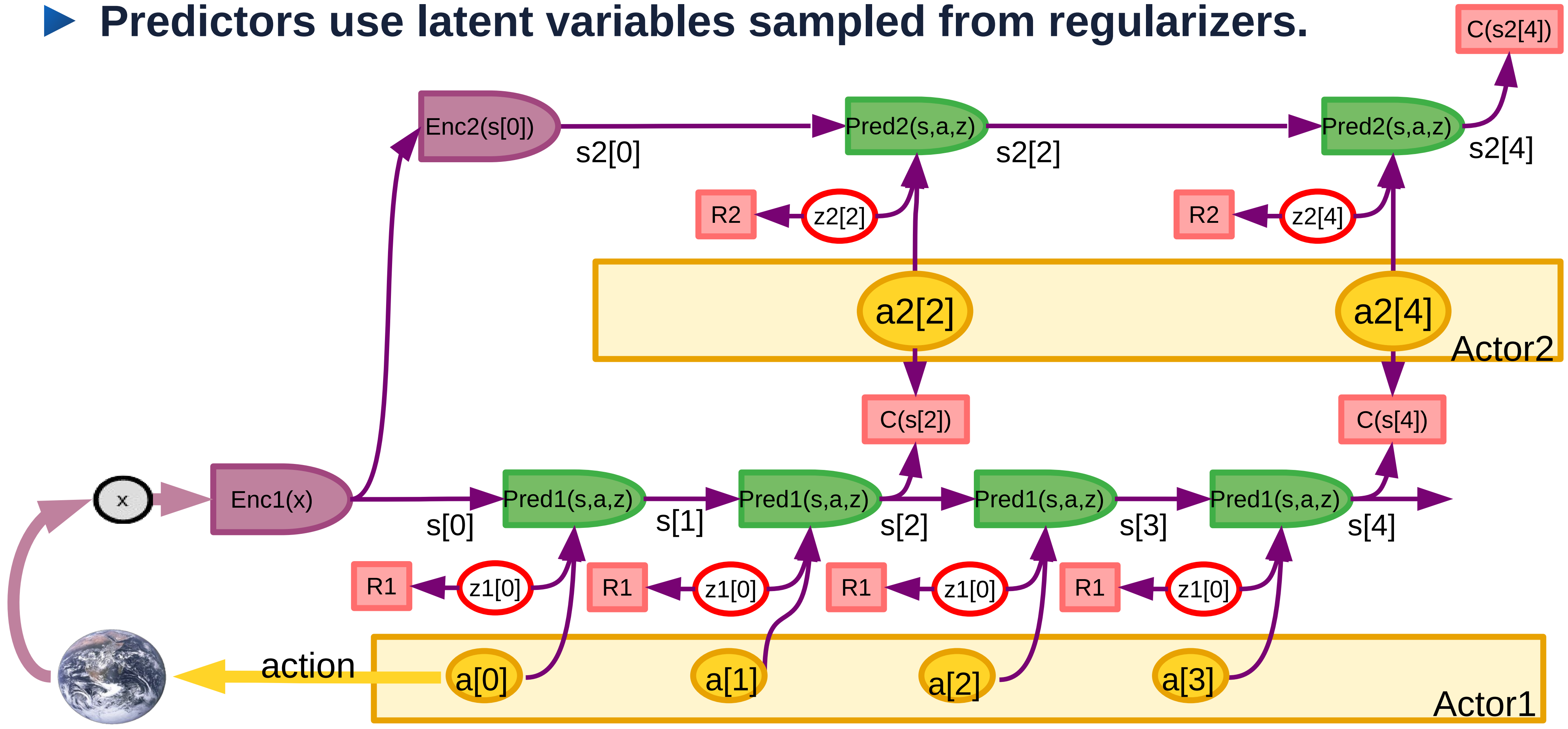
Mode-2 Planning with H-JEPA-based World Model

- **High level sets conditions from low level to satisfy.**



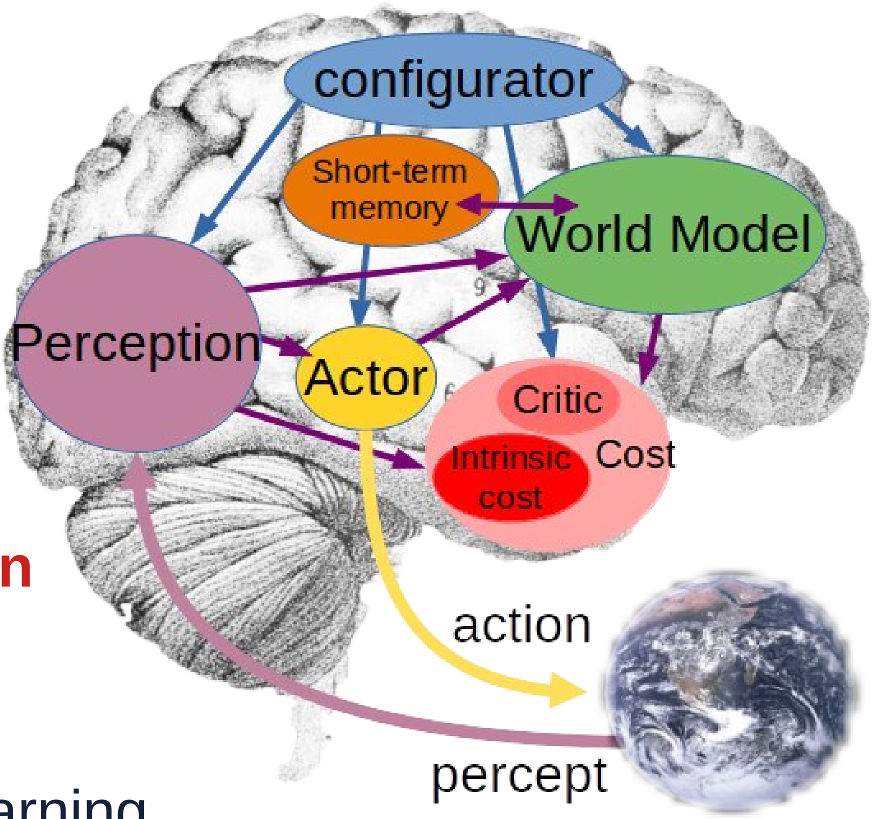
Hierarchical Planning with Uncertainty

- **Predictors use latent variables sampled from regularizers.**



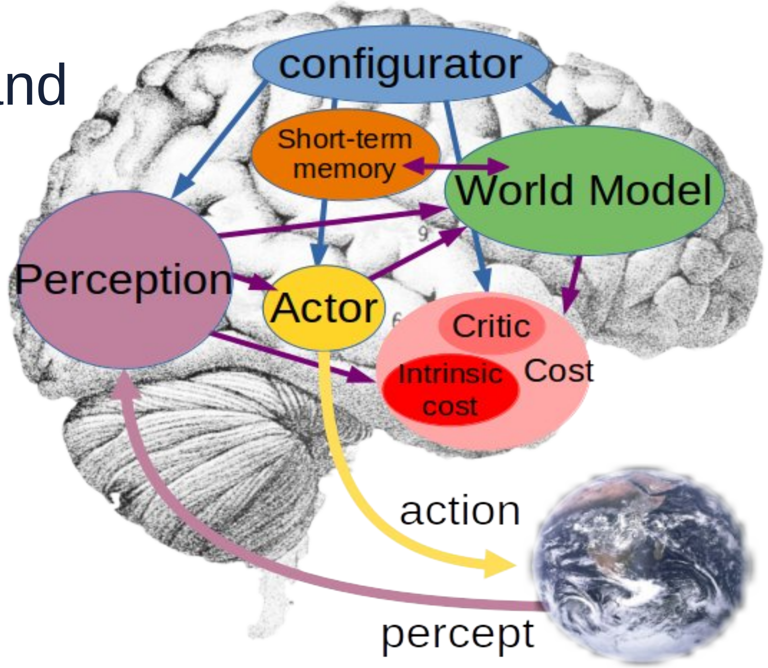
Steps towards Autonomous AI Systems

- **Self-Supervised Learning**
  - To learn representations of the world
  - To learn predictive models of the world
- **Handling uncertainty in predictions**
  - Joint-embedding predictive architectures
  - Energy-Based Model framework
- **Learning world models from observation**
  - Like animals and human babies?
- **Reasoning and planning**
  - That is compatible with gradient-based learning
  - No symbols, no logic → vectors & continuous functions



A Single, Configurable World Model Engine

- **What is the Configurator?**
- **The configurator configures the agent for a deliberate ("conscious") tasks.**
  - Configures all other modules for the task at hand
  - Primes the perception module
  - Provides executive control
  - Sets subgoals
  - Configures the world model for the task.



- **There is a single world model engine**
  - The system can only perform one "conscious" task at a time
  - Consciousness is a consequence of the single-world-model limitation

Positions / Conjectures / Crazy hypotheses

- **Prediction is the essence of intelligence**
  - Learning predictive models of the world is the basis of common sense
- **Almost everything is learned**
  - Low-level features, space, objects, physics, abstract representations…
- **H-JEPA with non-contrastive training are the thing**

  Probabilistic generative models and contrastive methods are doomed.
- **Almost everything is learned through self-supervised learning**
  - Almost nothing is learned through reinforcement, supervised or imitation
- **Intrinsic costs and architecture drive behavior & determine what we learn.**
- **Emotions are necessary for autonomous intelligence**
  - They are anticipation of outcomes by the critic or the world model+intrinsic cost.
- **Reasoning is an extension of simulation/prediction and planning.**
- **Consciousness exists because we have only one world model engine**

FACEBOOK AI

Thank You!