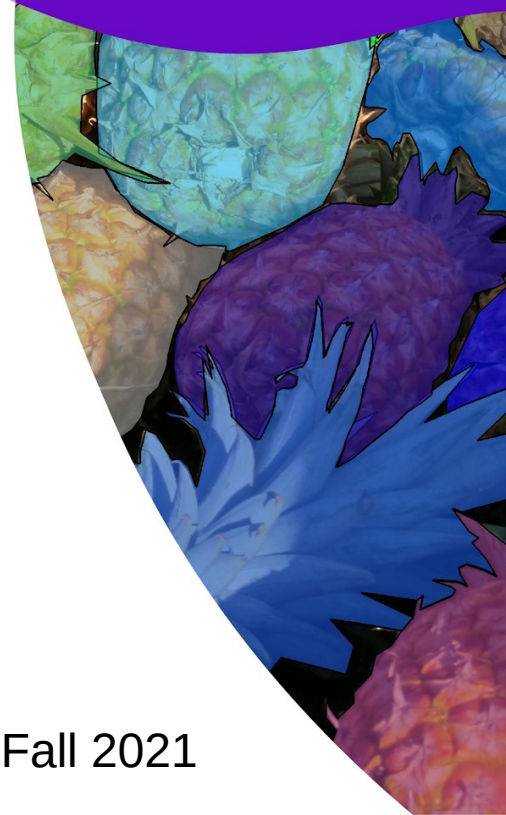
NEW YORK UNIVERSITY

# Energy-Based Models (part 2)



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

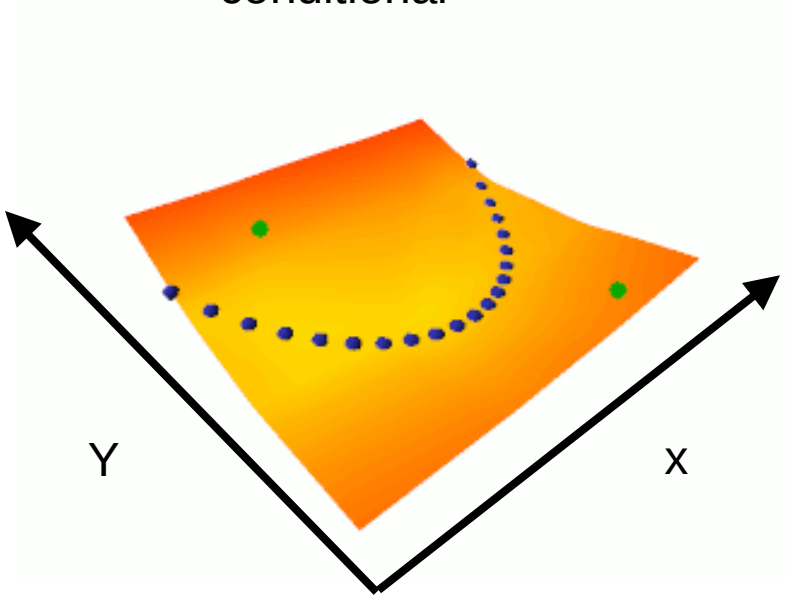
Deep Learning, NYU, Fall 2021

# Contrastive energy-based learning

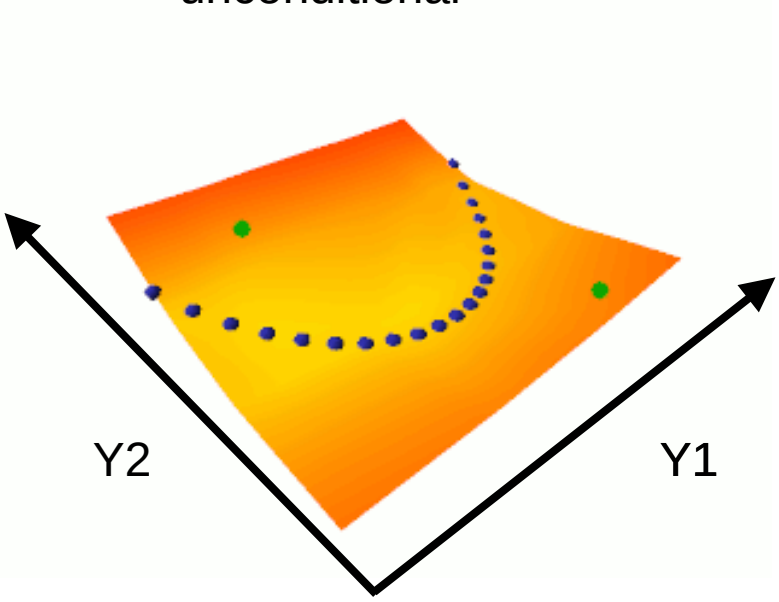
- **Push down on the energy of training samples**
- **Pull up on the energy of other “well-chosen” points**

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

conditional




unconditional

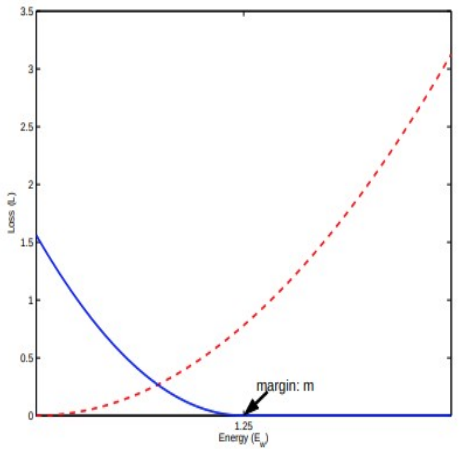


# Contrastive Methods vs Regularized/Architectural Methods

- **Contrastive: [they all are different ways to pick which points to push up]**
  - C1: push down of the energy of data points, push up everywhere else: Max likelihood (needs tractable partition function or variational approximation)
  - C2: push down of the energy of data points, push up on chosen locations: max likelihood with MC/MMC/HMC, Contrastive divergence, Metric learning/Siamese nets, Ratio Matching, Noise Contrastive Estimation, Min Probability Flow, adversarial generator/GANs
  - C3: train a function that maps points off the data manifold to points on the data manifold: denoising auto-encoder, masked auto-encoder (e.g. BERT)
- **Regularized/Architectural: [Different ways to limit the information capacity of the latent representation]**
  - A1: build the machine so that the volume of low energy space is bounded: PCA, K-means, Gaussian Mixture Model, Square ICA, normalizing flows…
  - A2: use a regularization term that measures the volume of space that has low energy: Sparse coding, sparse auto-encoder, LISTA, Variational Auto-Encoders, discretization/VQ/VQVAE.
  - A3: F(x,y) = C(y, G(x,y)), make G(x,y) as "constant" as possible with respect to y: Contracting auto-encoder, saturating auto-encoder
  - A4: minimize the gradient and maximize the curvature around data points: score matching

# Contrastive losses: pairwise margin losses

- **Push down on data points, push up of other points**
  - well chosen contrastive points
  - General margin loss: $\mathcal{L}(x, y, \hat{y}, w) = H(F_w(x, y), F_w(x, \hat{y}), m(y, \hat{y}))$
    - Where H($F^+$,$F^-$,m) is a strictly increasing function of $F^+$ and a strictly decreasing function of $F^-$ , at least whenever $F^- - F^+ < m$.
  - Examples:
    - Simple [Bromley 1993]:
      $\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y)]^+ + [m(y, \hat{y}) - F_w(x, \hat{y})]^+$
    - Hinge pair loss [Altun 2003], Ranking loss [Weston 2010]:
      $\mathcal{L}(x, y, \hat{y}, w) = [F_w(x, y) - F_w(x, \hat{y}) + m(y, \hat{y})]^+$
    - Square-Square: [Chopra CVPR 2005] [Hadsell CVPR 2006]:
      $\mathcal{L}(x, y, \hat{y}, w) = \left([F_w(x, y)]^+\right)^2 + \left([m(y, \hat{y}) - F_w(x, \hat{y})]^+\right)^2$



# Plenty of Contrastive Loss Functions to Choose From

## Good and bad loss functions: good ones have non-zero margin

| Loss (equation #) | Formula | Margin |
|---|---|---|
| energy loss | $E(W, Y^i, X^i)$ | none |
| perceptron | $E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$ | 0 |
| hinge | $\max\left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$ | $m$ |
| log | $\log\left(1 + e^{E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)}\right)$ | $> 0$ |
| LVQ2 | $\min\left(M, \max(0, E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\right)$ | 0 |
| MCE | $\left(1 + e^{-\left(E(W,Y^i,X^i) - E(W,\bar{Y}^i,X^i)\right)}\right)^{-1}$ | $> 0$ |
| square-square | $E(W, Y^i, X^i)^2 - \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2$ | $m$ |
| square-exp | $E(W, Y^i, X^i)^2 + \beta e^{-E(W,\bar{Y}^i,X^i)}$ | $> 0$ |
| NLL/MMI | $E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$ | $> 0$ |
| MEE | $1 - e^{-\beta E(W,Y^i,X^i)} / \int_{y \in \mathcal{Y}} e^{-\beta E(W,y,X^i)}$ | $> 0$ |

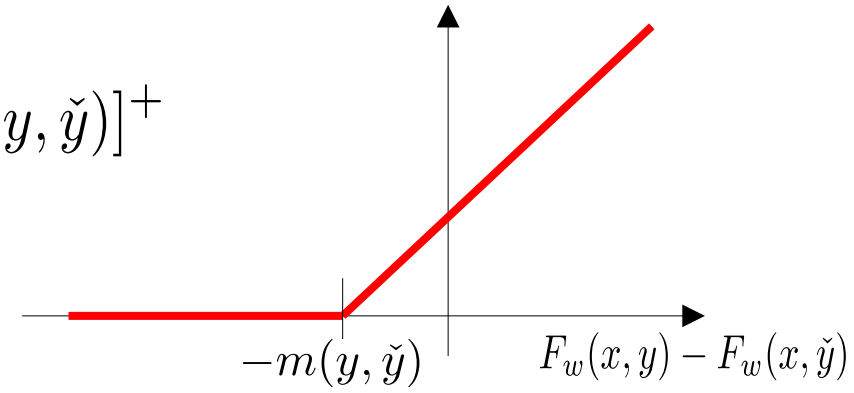
# General additive margin loss

- **Considers all possible outputs**

$$\mathcal{L}(x, y, w) = \sum_{\check{y} \in \mathcal{Y}} H(F_w(x, y), F_w(x, \check{y}), m(y, \check{y}))$$

- **Hinge loss that makes F(x,y) lower than F(x,y') by a quantity (margin) that depends on the distance between y and y'**
- **Example:**

$$\mathcal{L}(x, y, w) = \sum_{\check{y} \in \mathcal{Y}} [F_w(x, y) - F_w(x, \check{y}) + m(y, \check{y})]^+$$



# Contrastive losses: competitive group losses

- **Push down on data points, push up of well-chosen other points**

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

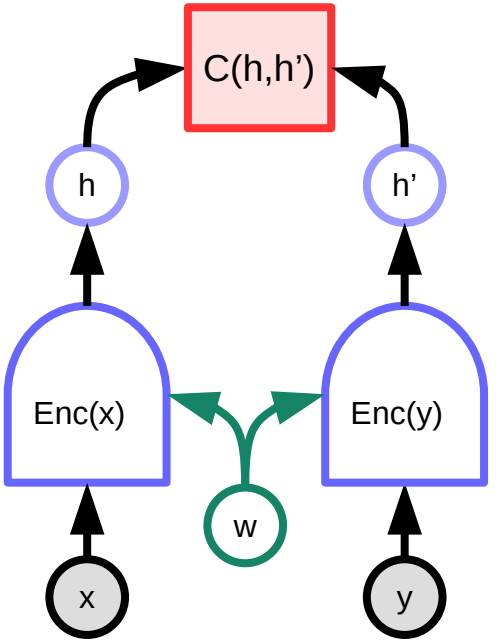
- H: increasing function of F(x,y), decreasing function of F(x,ŷ) while the difference is larger than -M()

- Example: InfoNCE, Contrastive Predictive Coding [van den Oord 2018]
  - Used a lot for Siamese net and joint embedding
  - Margin is implicit and infinite
  - Contrastive samples compete for gradient

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

# Contrastive Embedding

- **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: hard negative mining**
- **Successful examples for images:**
  - DeepFace [Taigman et al. CVPR’14]
  - PIRL [Misra et al. Arxiv:1912.01991]
  - MoCo v1, v2 [He et al. Arxiv:1911.05722]
  - SimCLR v1, v2 [Chen et al. ArXiv:2002.05709]
- **Speech:**
  - Wav2vec 2.0 [Baevski et al. ArXiv:2006.11477]

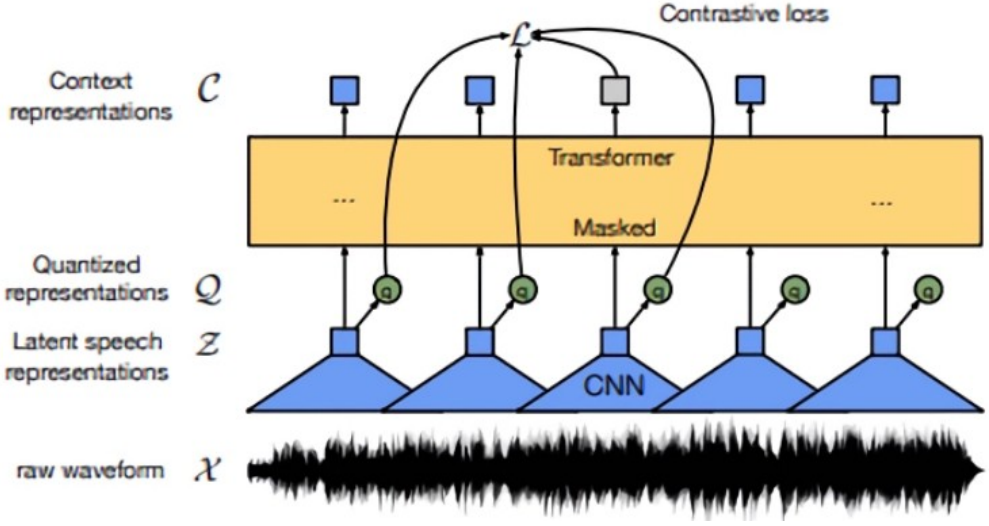


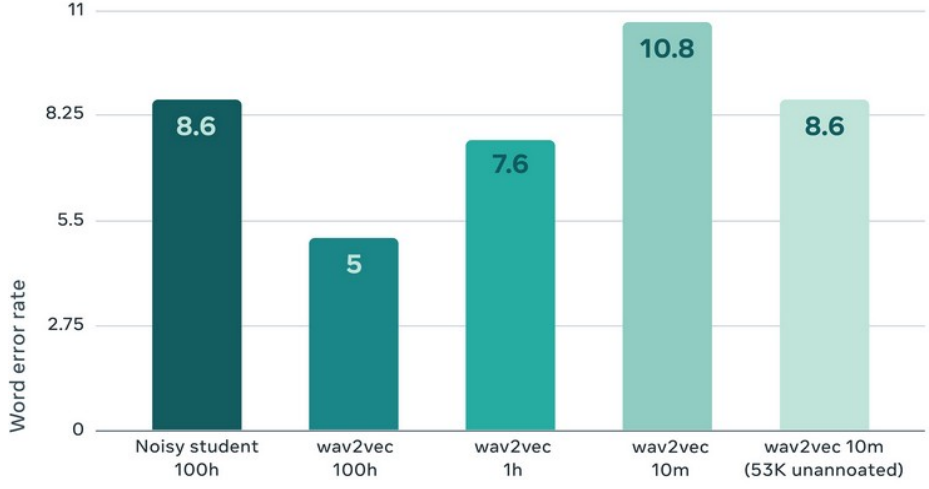

Positive pair:
Make F small

Negative pair:
Make F large

# 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

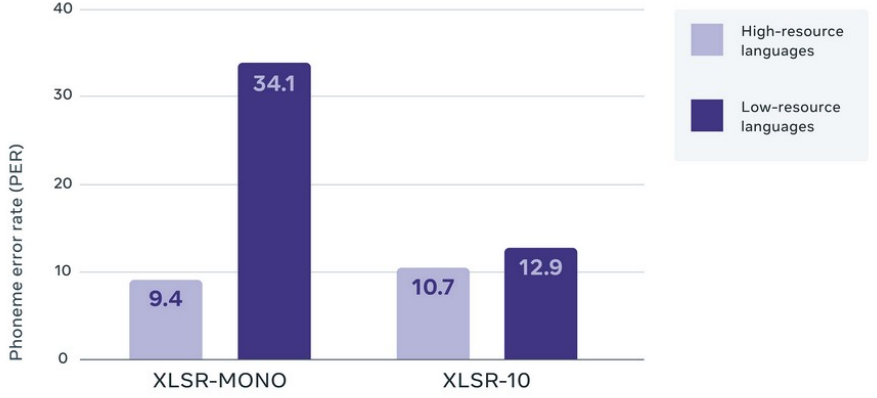


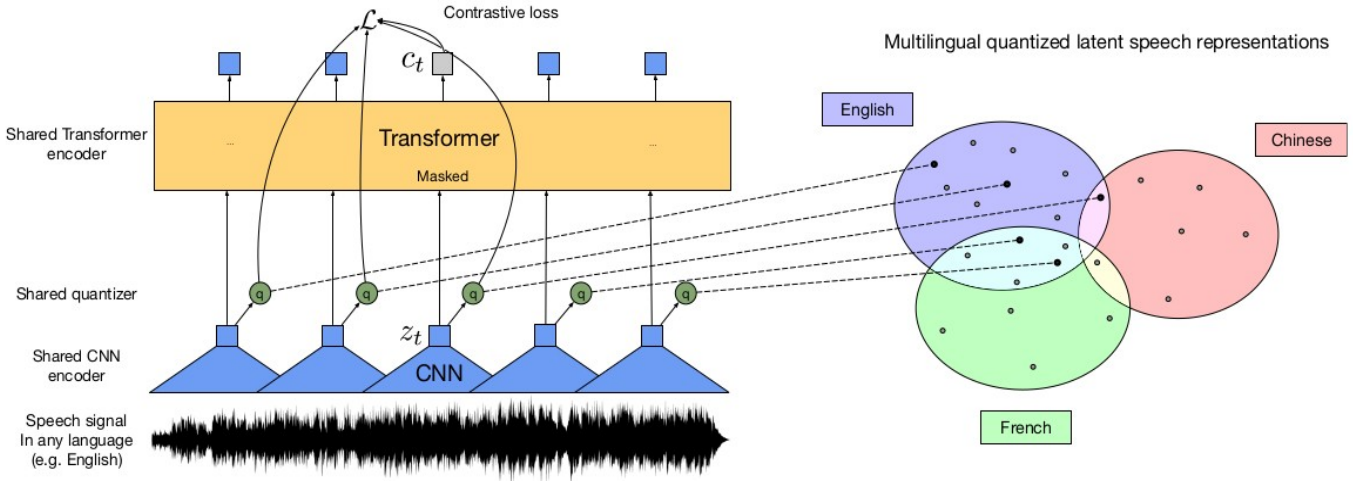

WER for Noisy Student self-training with 100 hours of labeled data. Wav2vec 2.0 with 100 hours, 1 hour, and only 10 minutes of labeled data. All models use the remainder of the LibriSpeech corpus (total 960 hours) as unannotated data, except for the last result, which uses 53K hours from LibriVox.

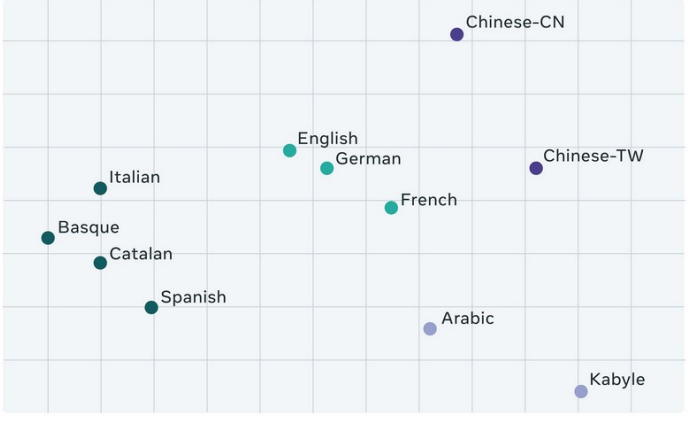
# XLSR: multilingual speech recognition

- **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).



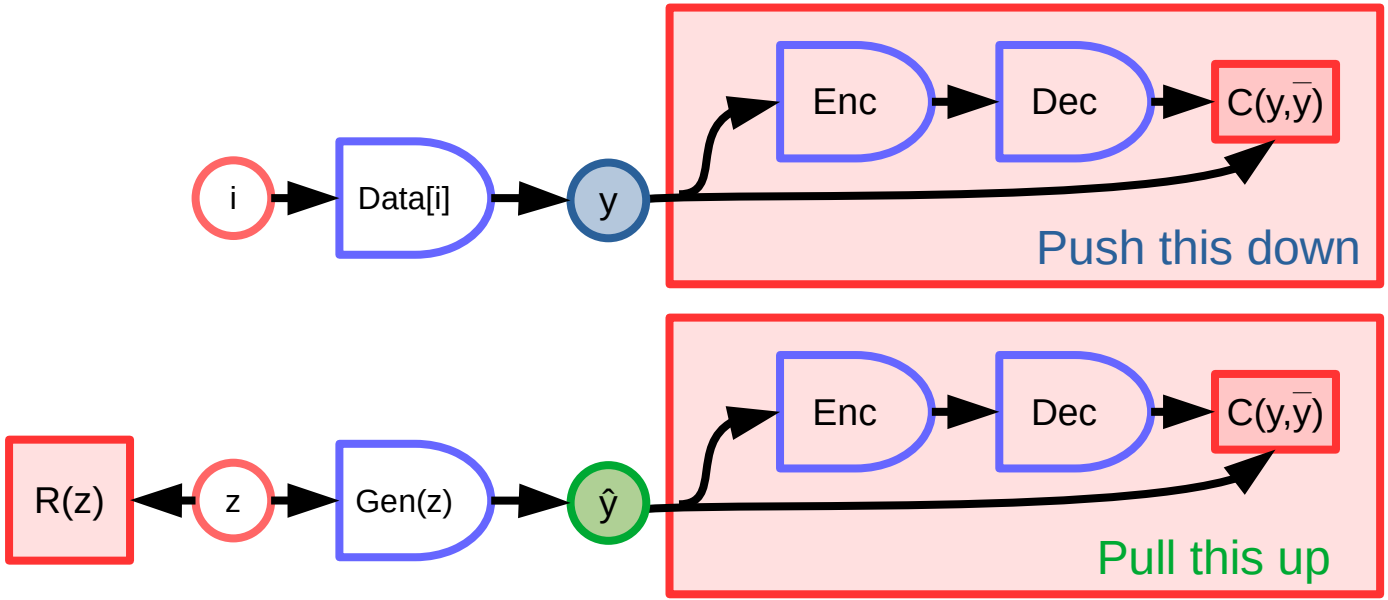



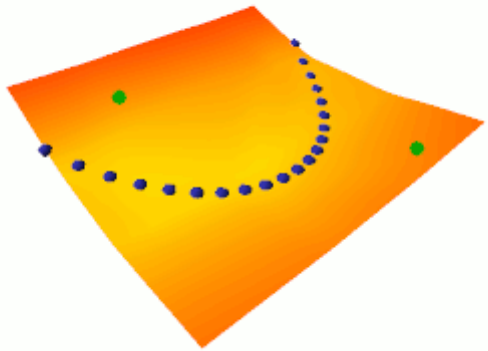

Visualization of how the learned units are used across languages. Graph shows a 2D PCA plot of how units are used in each language. Languages closer to each other, like English and German or Basque and Catalan, tend to use similar units.

# GANs are secretly a contrastive method for EBM

- **Energy-Based GAN** [Zhao 2016]**, Wasserstein GAN** [Arjovsky 2017]**,...**
  - GANs generate nice images
  - But learning representations of image has not been very successful.

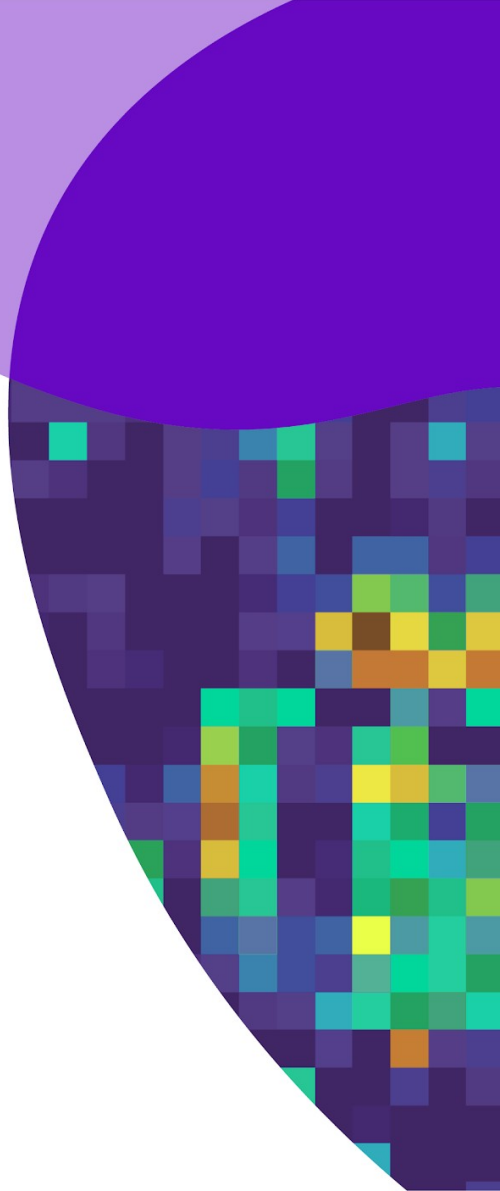
$$\mathcal{L}(x, y, \hat{y}, w) = H(F_w(x, y), F_w(x, \hat{y}), m(y, \hat{y}))$$



FACEBOOK AI

# Non-Contrastive Methods for Joint Embedding



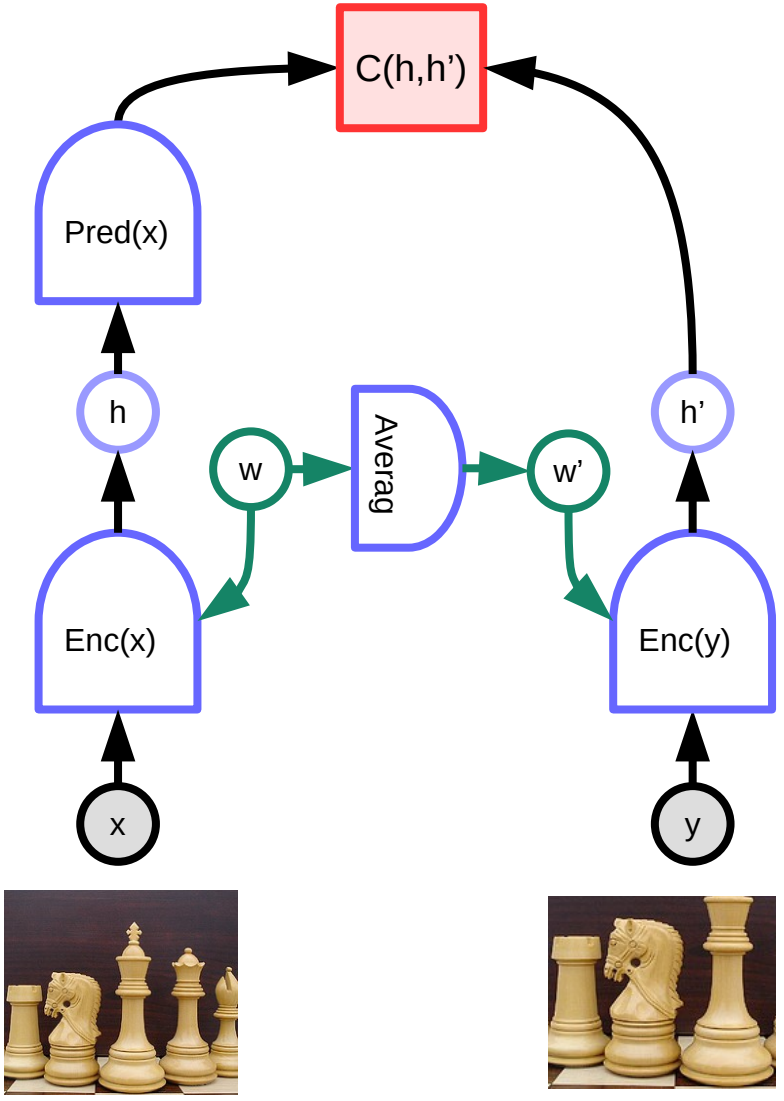
Other tricks to prevent collapse

# Non-Contrastive Embedding

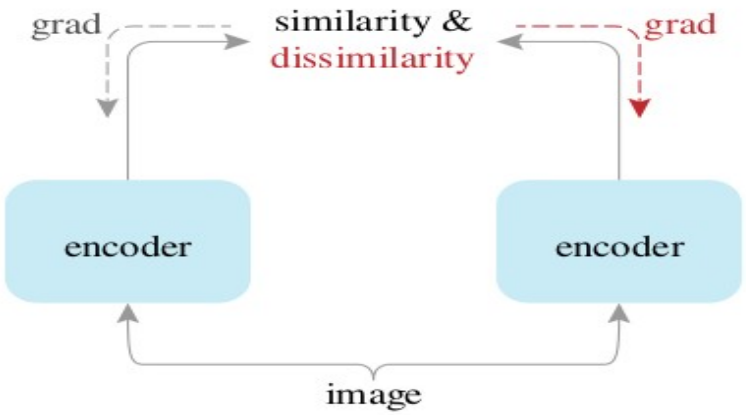
- **Advantage: no pixel-level reconstruction**
- **Eliminates hard negative mining**

- **Siamese nets with slightly different weights**
  - Bootstrap Your Own Latent (????)
    [Grill arXiv:2006.07733]
  - SimSiam [Chen arXiv:2011.10566]

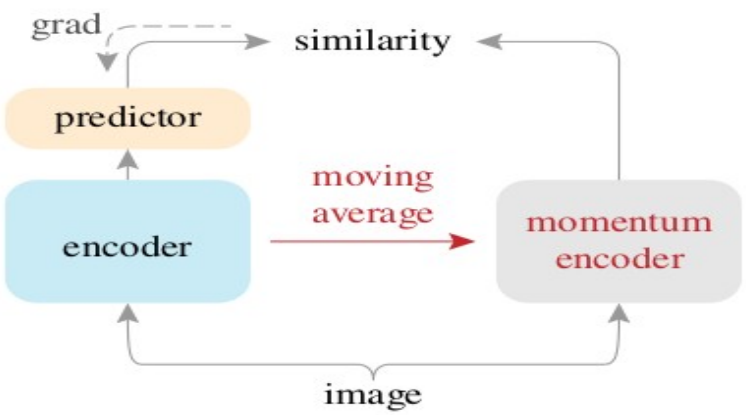
- **Using cluster centers as targets**
  - DeepCluster [Caron arXiv:1807.05520]
  - SwAV [Caron arXiv:2006.09882]



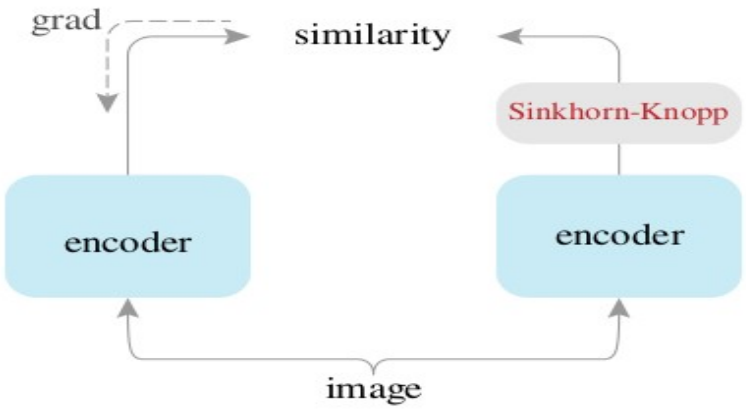
# Joint Embedding Methods



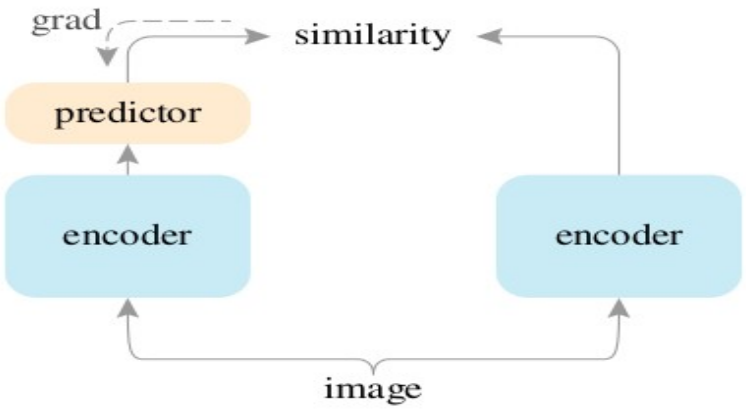

SimCLR




BYOL




SwAV

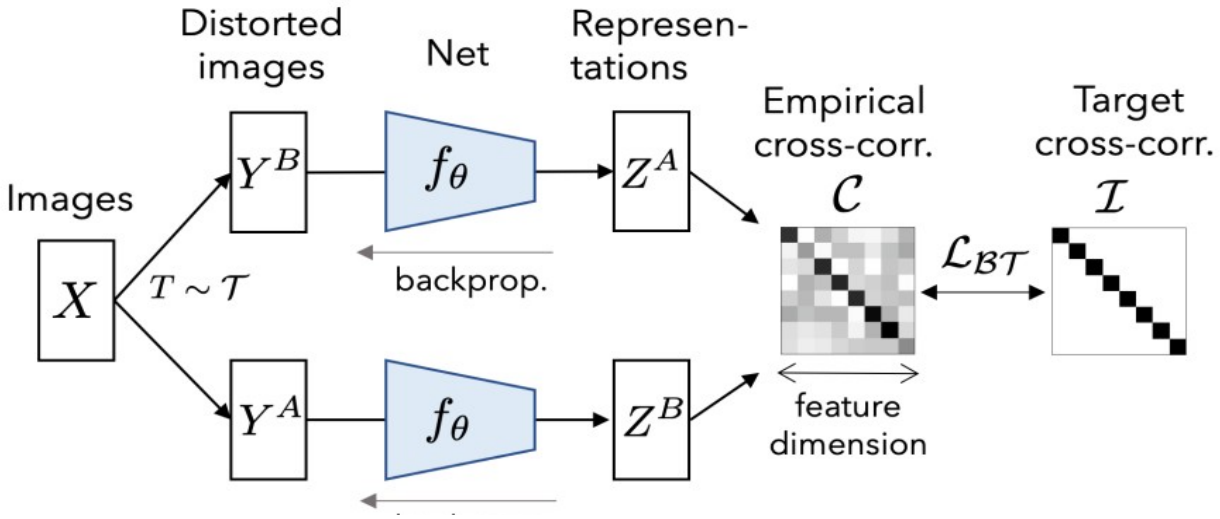



SimSiam

[Chen arXiv:2011.10566])

# Barlow Twins [Zbontar et al. ArXiv:2103.03230]

- Accuracy on ImageNet with linear classifier head
- Best accuracy for d=16,384 and batch-size=1024




| Method | Top-1 | Top-5 |
|---|---|---|
| Supervised | 76.5 | |
| MoCo | 60.6 | |
| PIRL | 63.6 | - |
| SimCLR | 69.3 | 89.0 |
| MoCo v2 | 71.1 | 90.1 |
| SimSiam | 71.3 | - |
| SwAV | 71.8 | - |
| BYOL | 74.3 | 91.6 |
| SwAV (w/ multi-crop) | 75.3 | - |
| Barlow Twins (ours) | 73.2 | 91.0 |

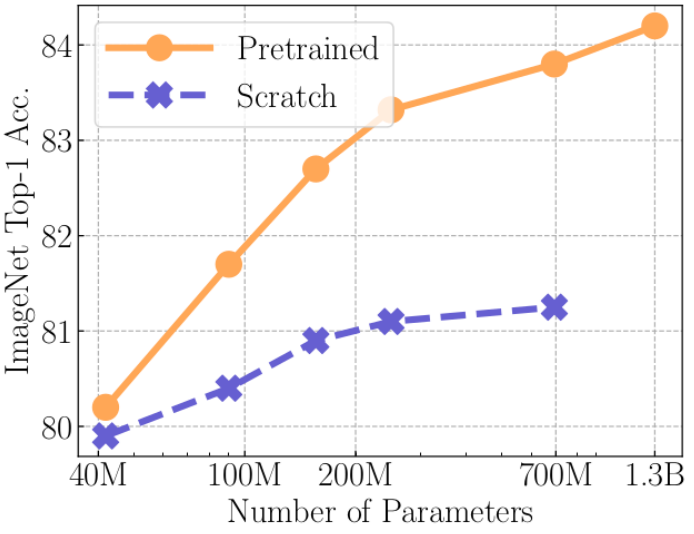
| Method | VOC07+12 det | | | COCO det | | | COCO instance seg | | |
|---|---|---|---|---|---|---|---|---|---|
| | $AP_{all}$ | $AP_{50}$ | $AP_{75}$ | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ | $AP^{mk}$ | $AP^{mk}_{50}$ | $AP^{mk}_{75}$ |
| Sup. | 53.5 | 81.3 | 58.8 | 38.2 | 58.2 | 41.2 | 33.3 | 54.7 | 35.2 |
| MoCo-v2 | **57.4** | 82.5 | **64.0** | **39.3** | 58.9 | **42.5** | **34.4** | 55.8 | 36.5 |
| SwAV | 56.1 | **82.6** | 62.7 | 38.4 | 58.6 | 41.3 | 33.8 | 55.2 | 35.9 |
| SimSiam | 57 | 82.4 | 63.7 | 39.2 | **59.3** | 42.1 | **34.4** | **56.0** | **36.7** |
| BT (ours) | 56.8 | **82.6** | 63.4 | 39.2 | 59.0 | **42.5** | 34.3 | **56.0** | 36.5 |

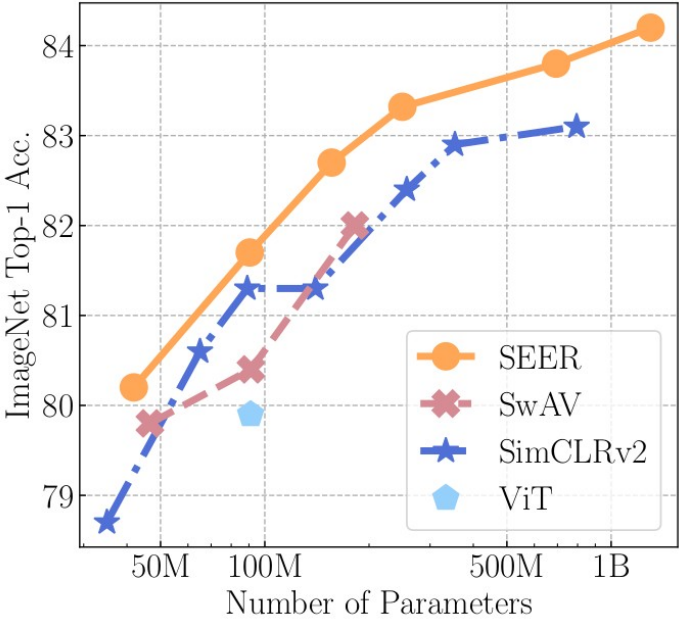
| Method | Top-1 | | Top-5 | |
|---|---|---|---|---|
| | 1% | 10% | 1% | 10% |
| Supervised | 25.4 | 56.4 | 48.4 | 80.4 |
| PIRL | - | - | 57.2 | 83.8 |
| SimCLR | 48.3 | 65.6 | 75.5 | 87.8 |
| BYOL | 53.2 | 68.8 | 78.4 | 89.0 |
| SwAV (w/ multi-crop) | 53.9 | **70.2** | 78.5 | **89.9** |
| Barlow Twins (ours) | **55.0** | 69.7 | **79.2** | 89.3 |

# 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** |

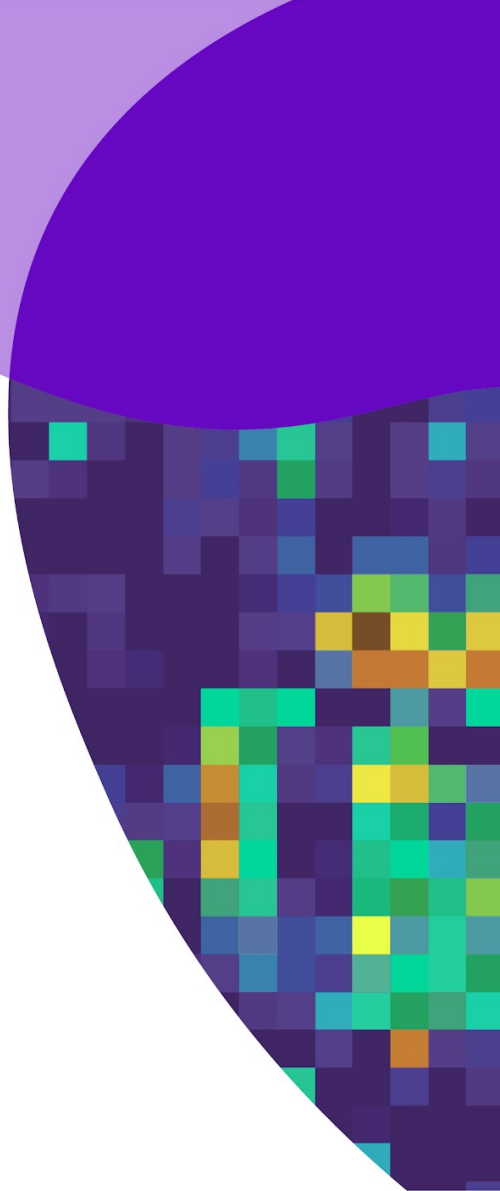


FACEBOOK AI

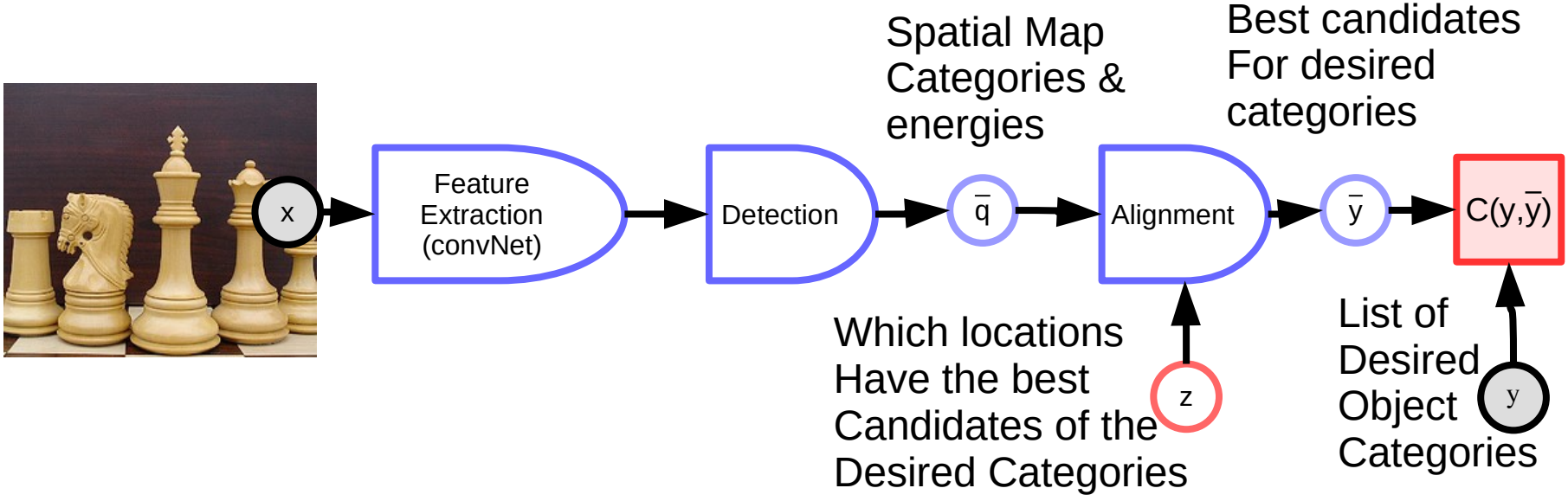
# Latent Variable Models in Practice

## Structured prediction with latent variable models



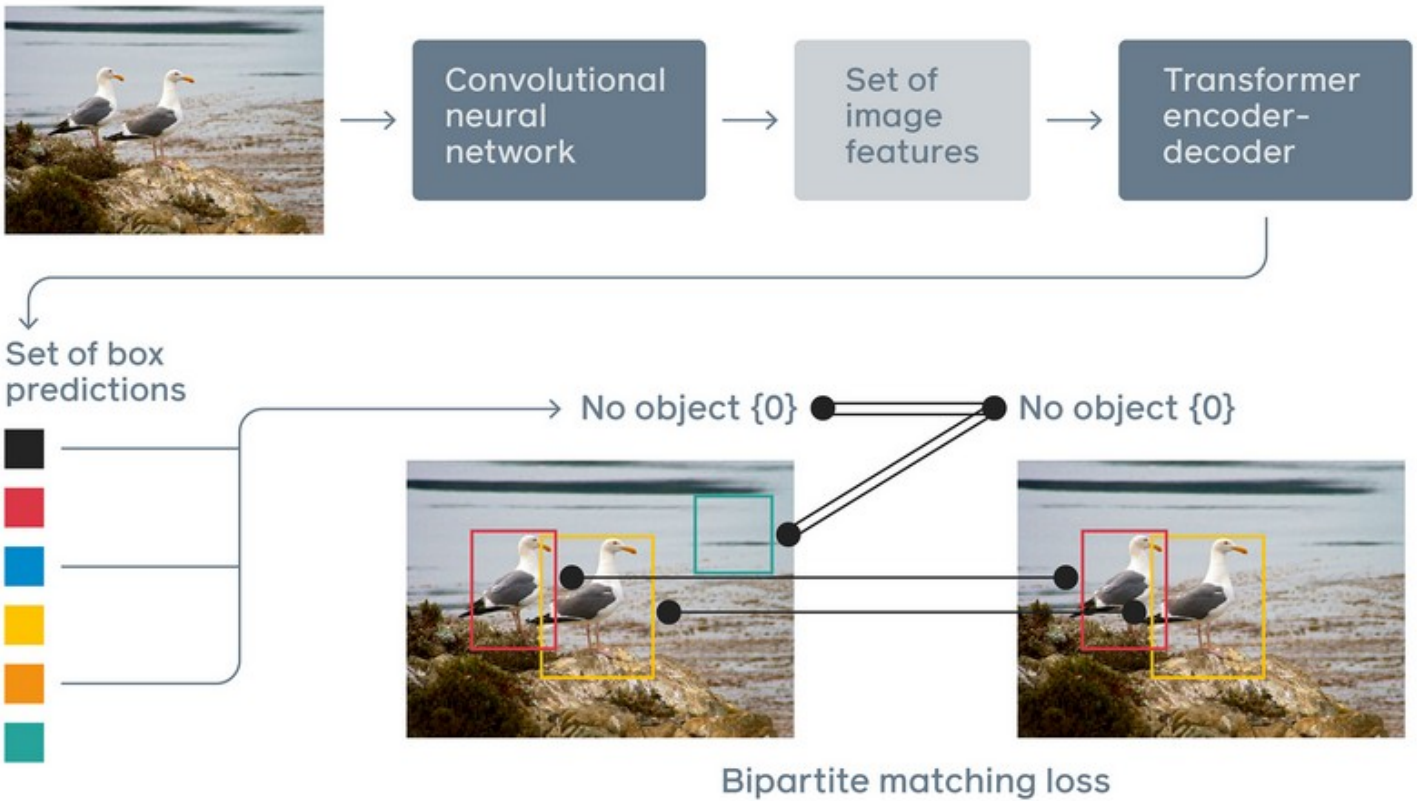
# Structured Prediction

- **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



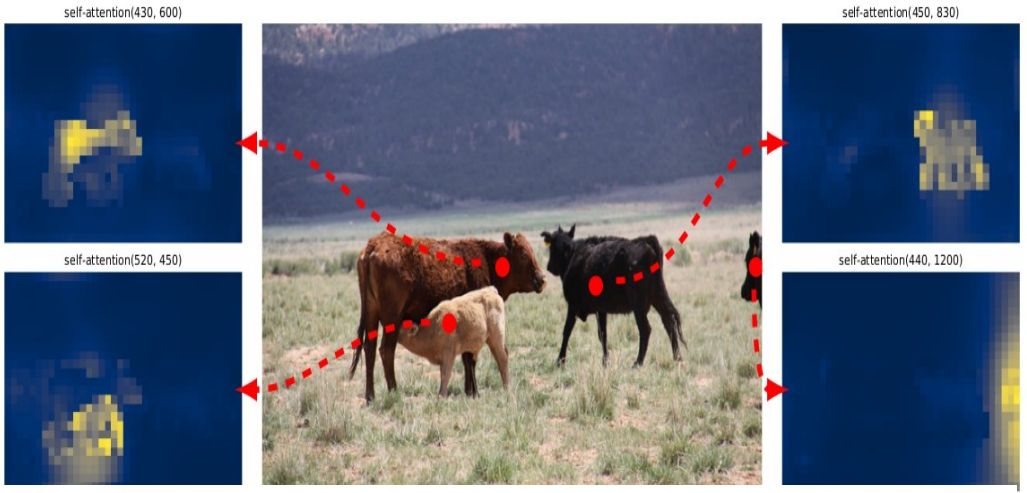
# DETR:

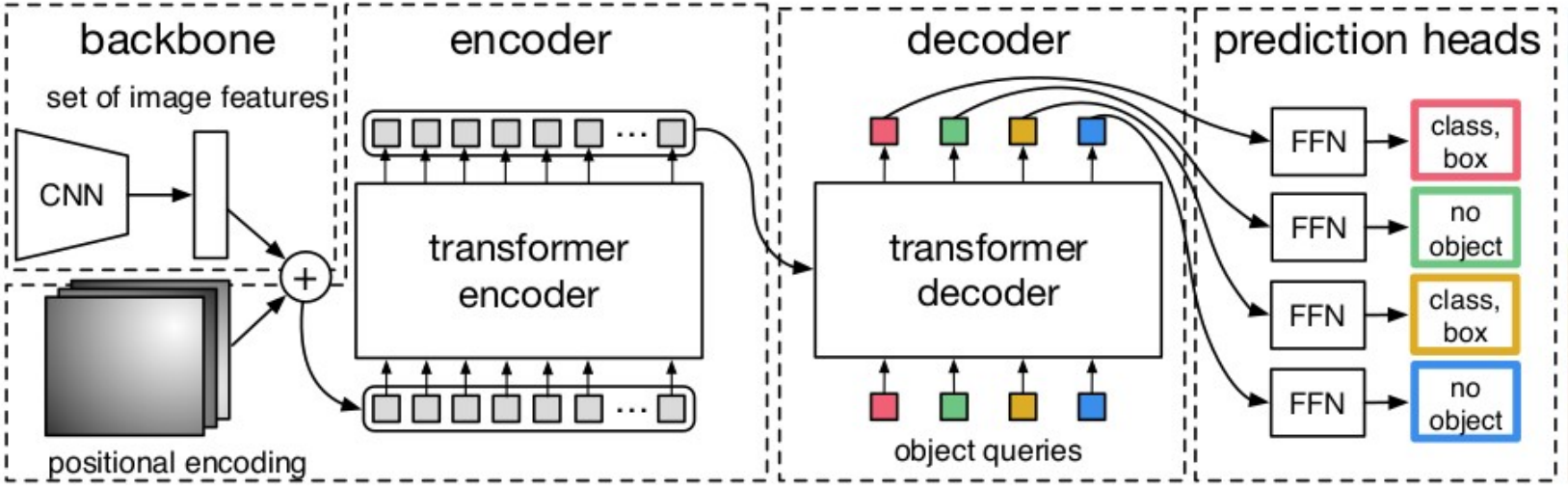
- DETR [Carion et al. ArXiv:2005.12872] https://github.com/facebookresearch/detr
- ConvNet → Transformer
- Object-based visual reasoning

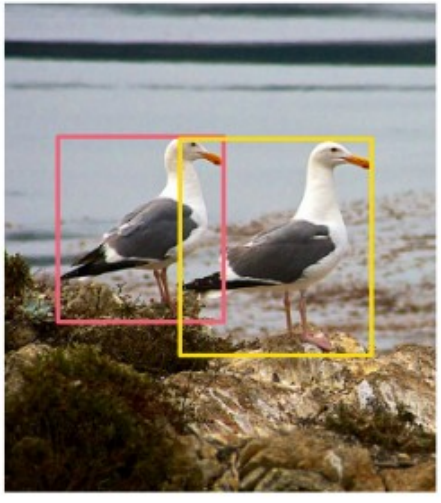


# DETR: ConvNet → Transformer for object detection

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

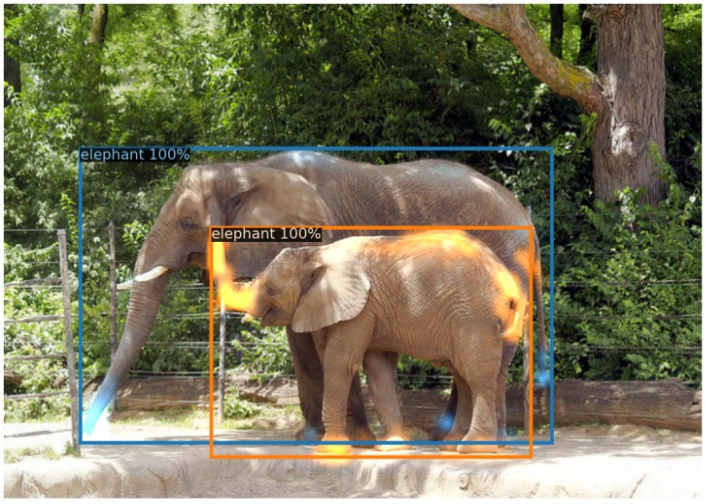


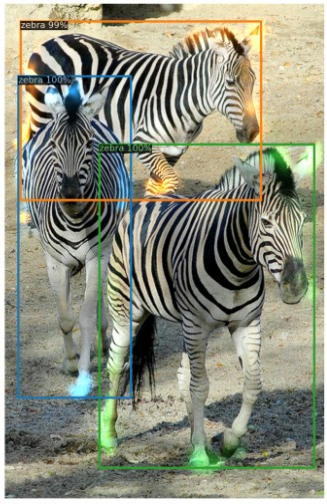



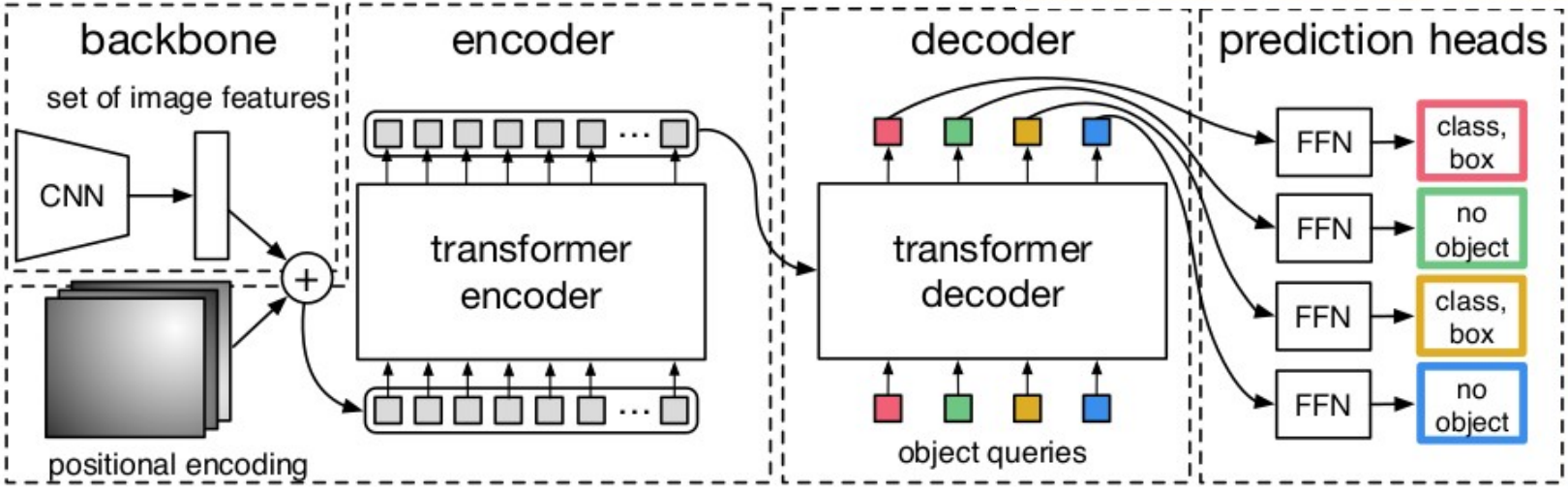



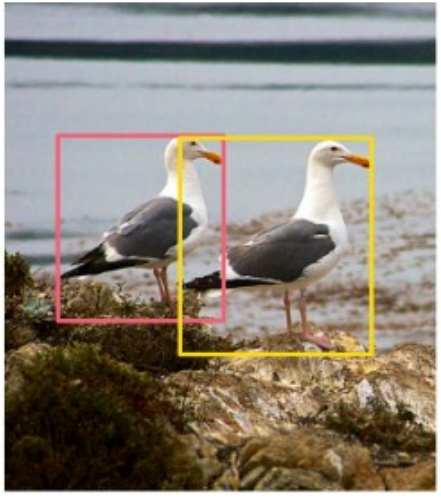
# DETR: ConvNet → Transformer for object detection

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

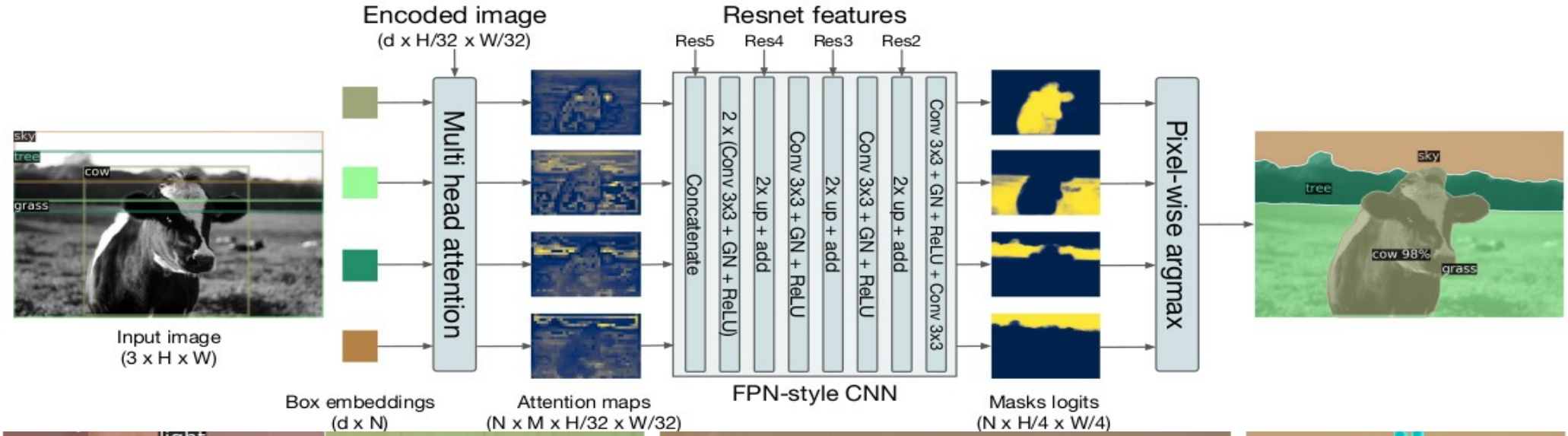


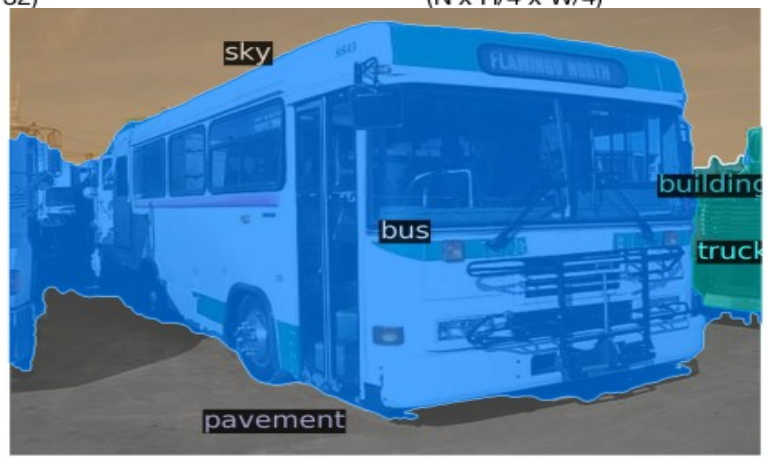



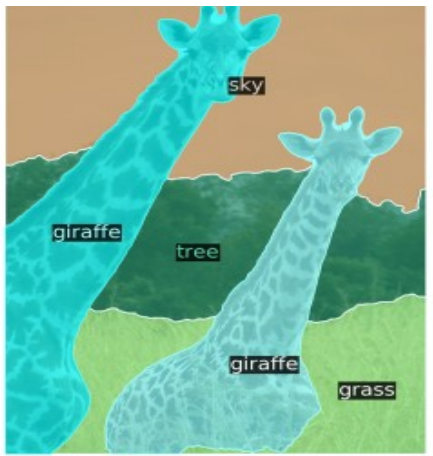



# DETR: results on panoptic segmentation





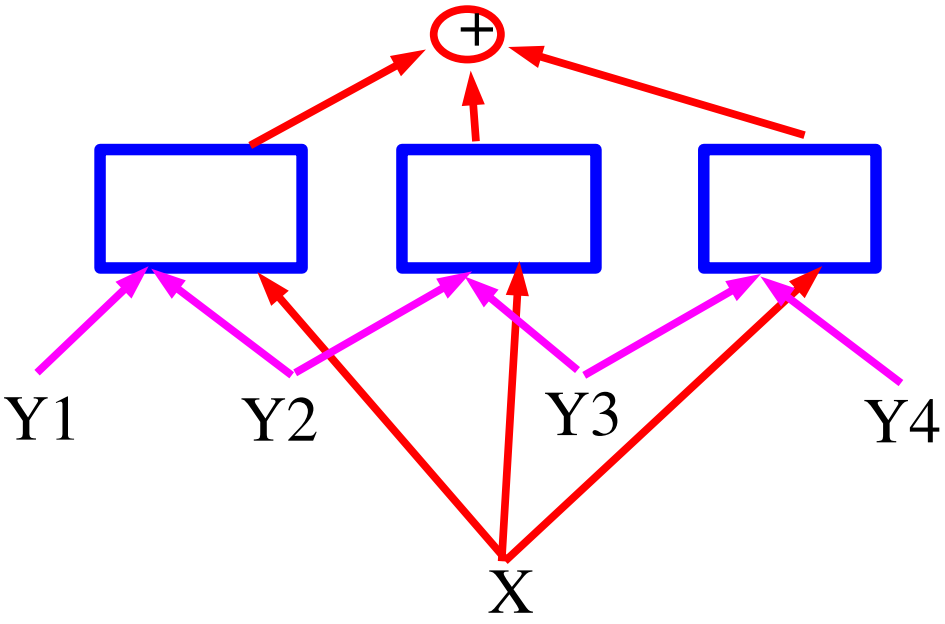


# Energy-Based Factor Graphs: Energy = Sum of "factors"

- **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)

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}, Z \in \mathcal{Z}} E(Z, Y, X).$$

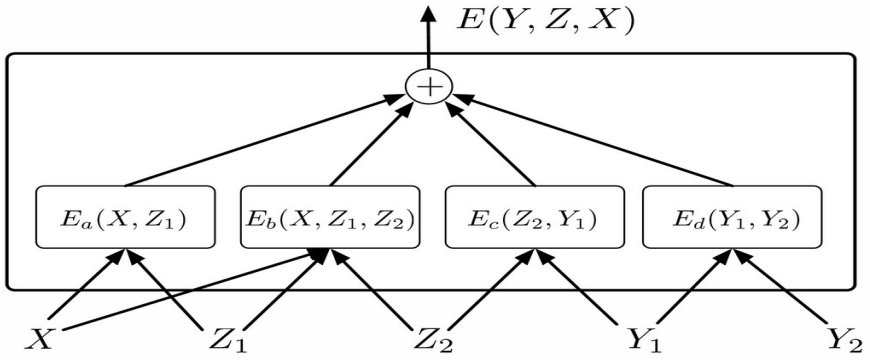


# Efficient Inference: Energy-Based Factor Graphs

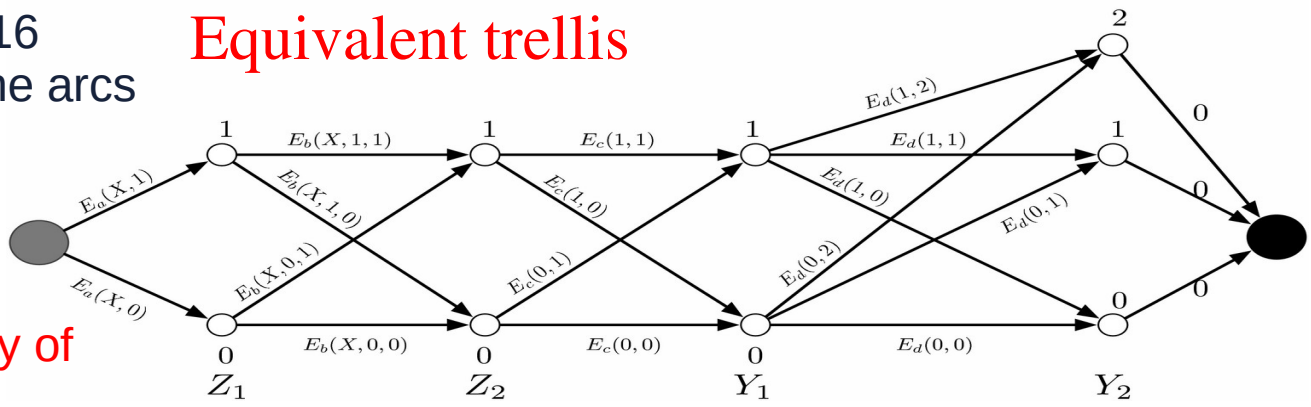
- **Example:**
  - Z1, Z2, Y1 are binary
  - Z2 is ternary
  - A naïve exhaustive inference would require 2x2x2x3=24 energy evaluations (= 96 factor evaluations)
  - BUT: Ea only has 2 possible input configurations, Eb and Ec have 4, and Ed 6.
  - Hence, we can precompute the 16 factor values, and put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

Factor graph

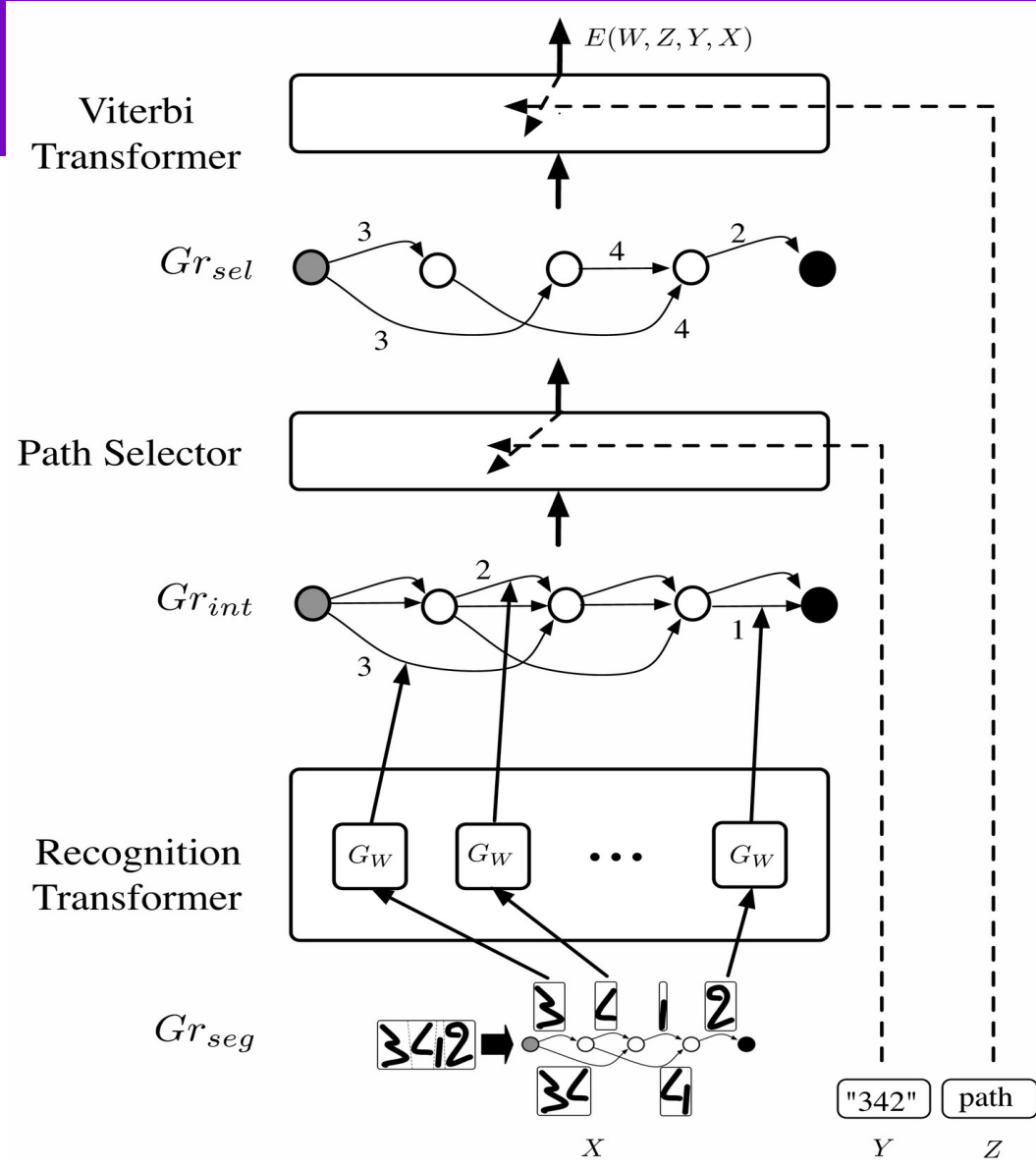



Equivalent trellis



# Deep Factors & implicit graphs: GTN

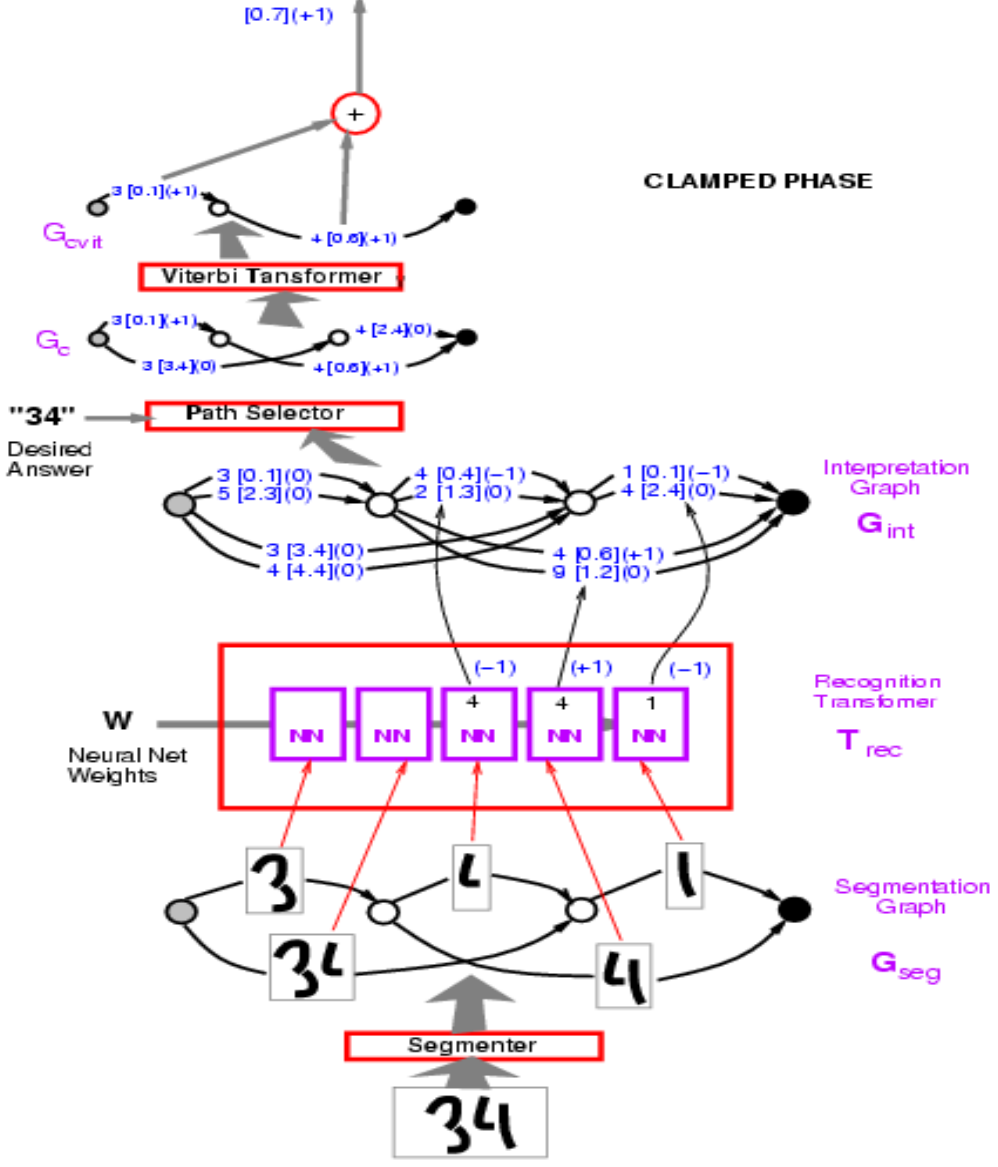
- **Handwriting Recognition with Graph Transformer Networks**
- **Un-normalized hierarchical HMMs**
  - Trained with Perceptron loss [LeCun, Bottou, Bengio, Haffner 1998]
  - Trained with NLL loss [Bengio, LeCun 1994], [LeCun, Bottou, Bengio, Haffner 1998]
- **Answer = sequence of symbols**
- **Latent variable = segmentation**



# Graph Transformer Networks

- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the energy of the desired answer:**

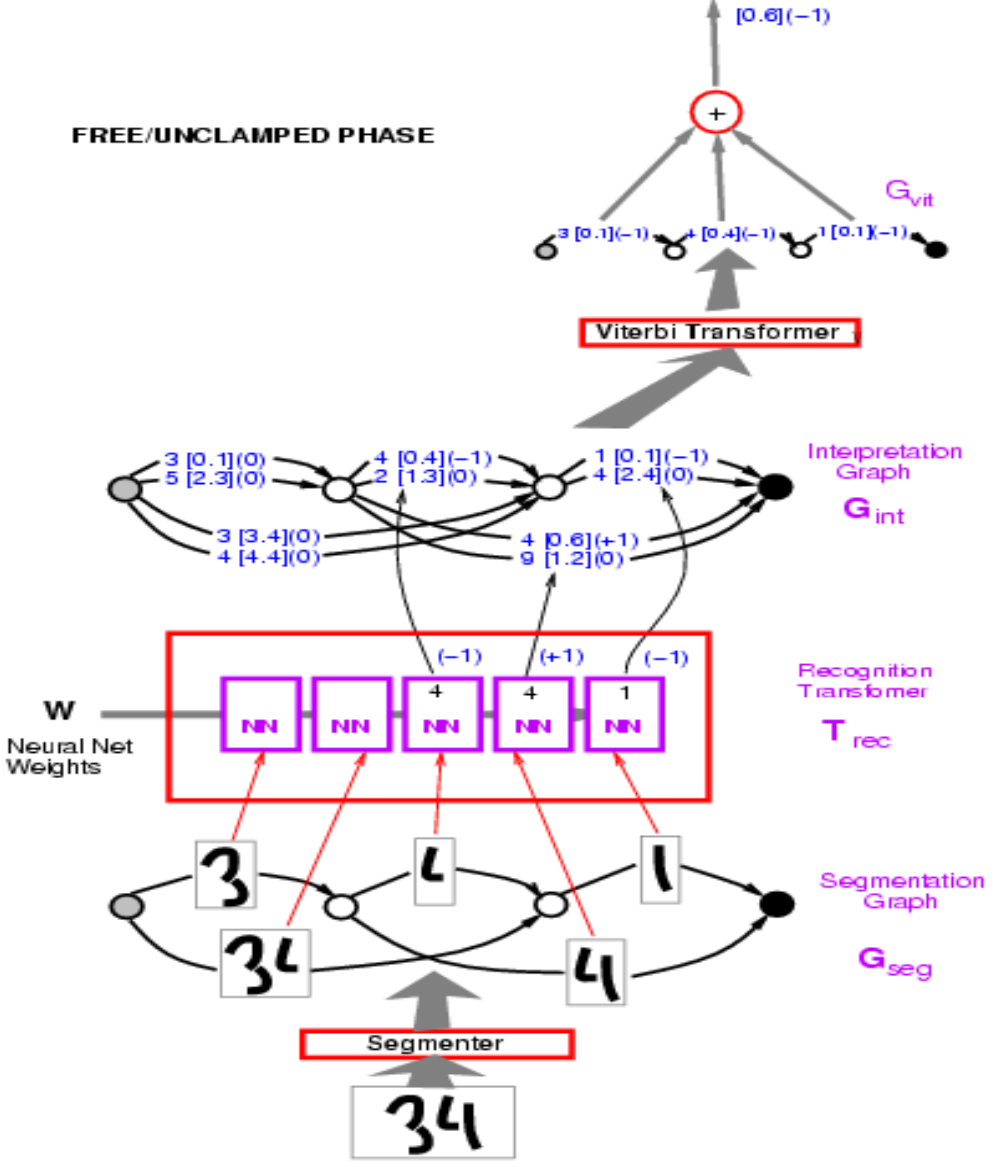
$$E(W, Y, X)$$



# Graph Transformer Networks

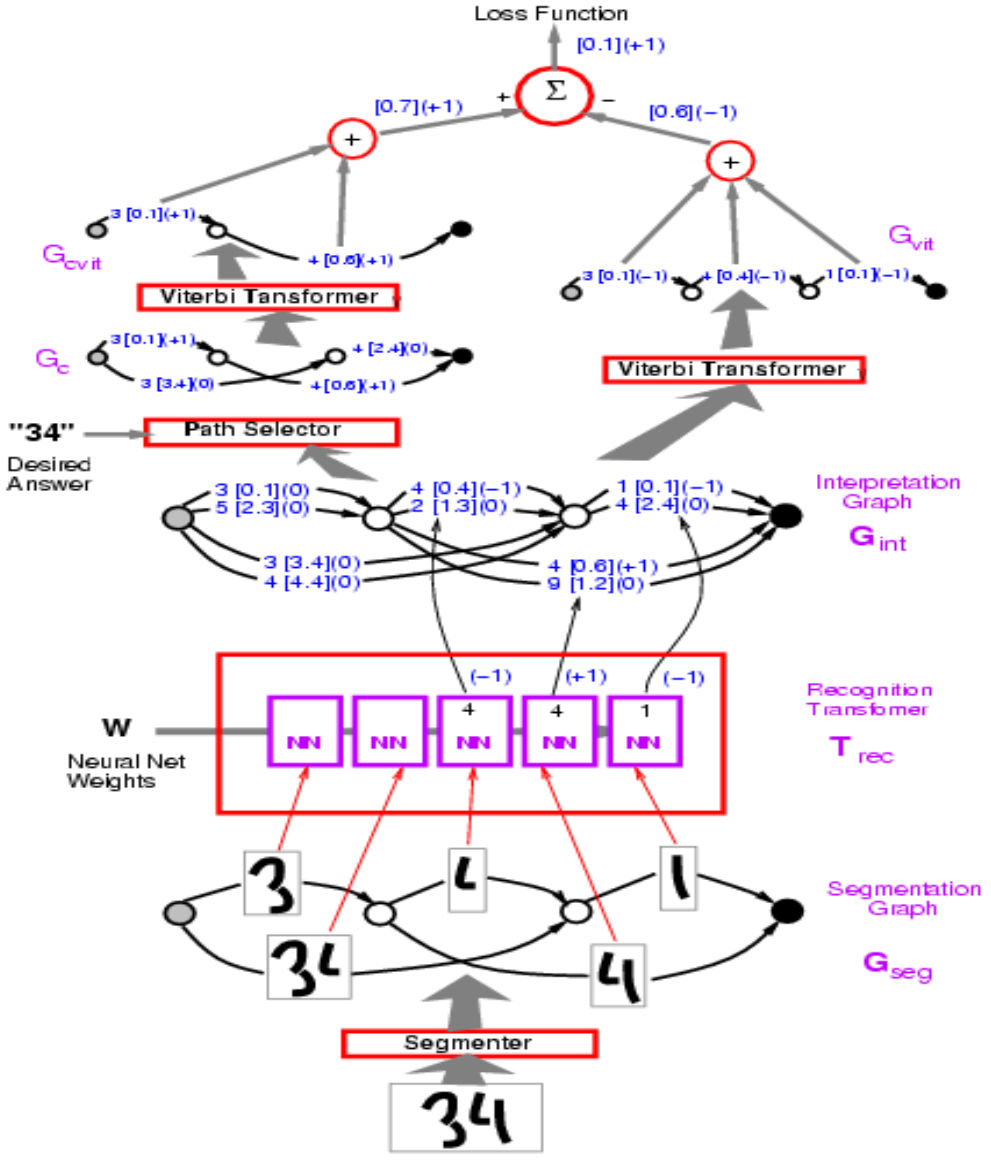
- **Variables:**
  - X: input image
  - Z: path in the interpretation graph/segmentation
  - Y: sequence of labels on a path
- **Loss function: computing the constrastive term:**

$$E(W, \check{Y}, X)$$



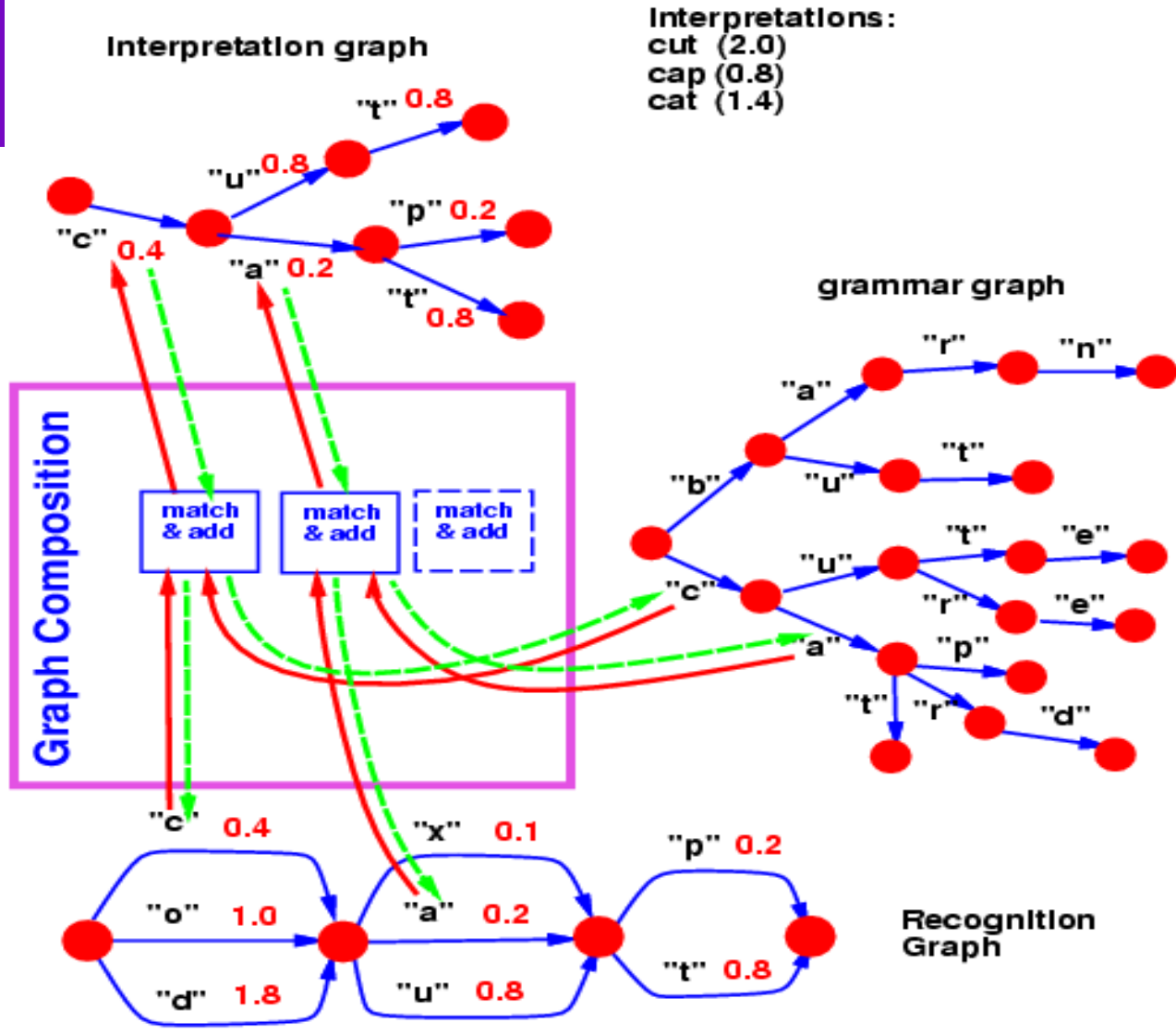
# Graph Transformer Networks

- **Example: Perceptron loss**
- **Loss = Energy of desired answer – Energy of best answer.**
  - (no margin)



# Graph Composition, Transducers.

- The composition of two graphs can be computed, the same way the dot product between two vectors can be computed.
- General theory: semi-ring algebra on weighted finite-state transducers and acceptors.



# Check Reader

- Graph transformer network trained to read **check amounts.**
- Trained globally with Negative-Log-Likelihood loss.
- 50% percent correct, 49% reject, 1% error (detectable later in the process.
- **Fielded in 1996**, used in many banks in the US and Europe.
- Processes an estimated **10% of all the checks written in the US in the late 1990s**