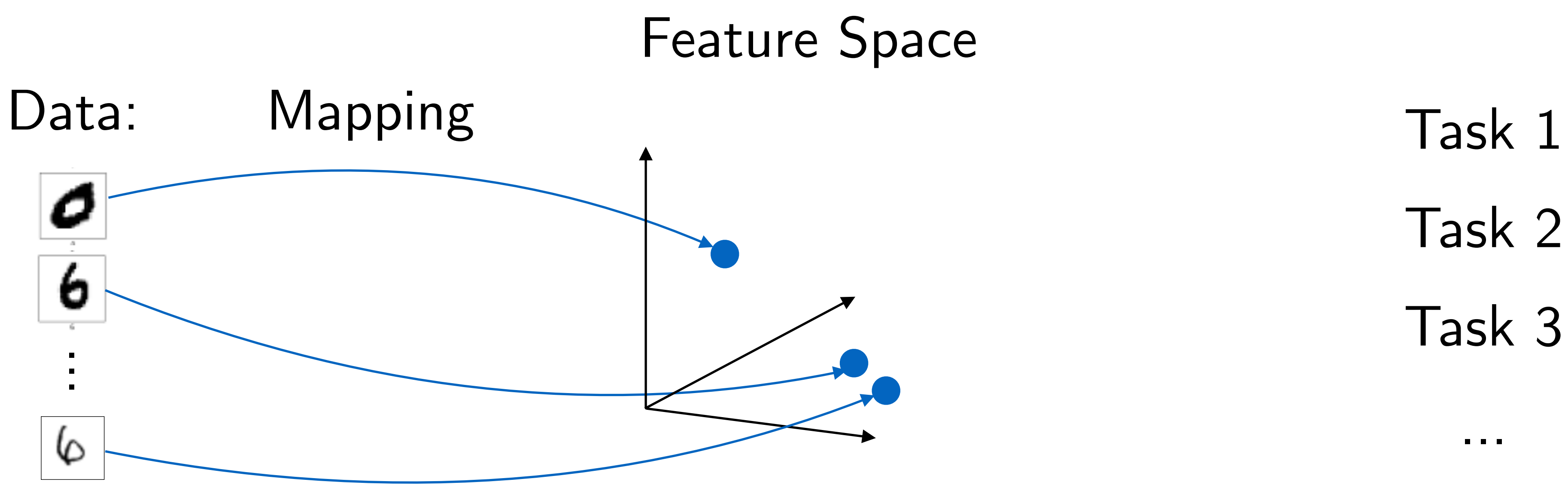
Deep Learning (BEV033DLE)
Lecture 10
Learning Representations I

Czech Technical University in Prague

✦ Lecture 10: LR-1:

- Feature Space Representations
- Word Vectors
- Similarity / Metric Learning
- Cross-Modality Representations

Feature Space Representation



✦ With good features many tasks are easy:

- E.g. logistic regression or kNN atop of deep features can perform very well (cf. fine-tuning experiments)
- Finding similar objects can be done by nearest neighbor search

✦ Suppose we are interested in high-level (semantic tasks). What would we like good features to do?

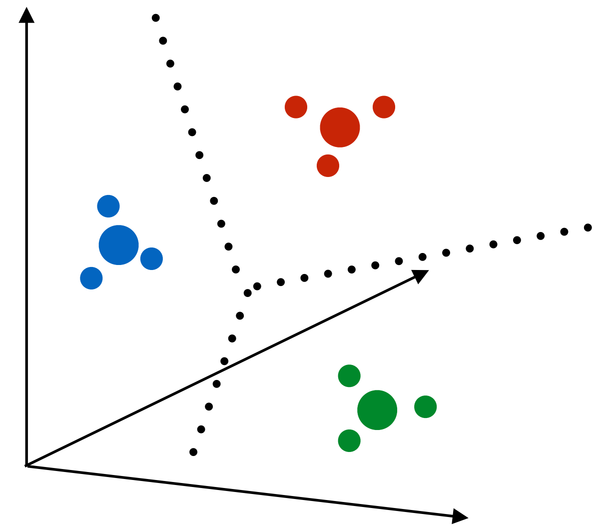
- Keep useful information (for all relevant tasks)
- Discard unnecessary information (view point, lighting, etc.). Or maybe separate?
- Meaningful metric in the feature space: similar representations should correspond to semantically similar objects (useful for many tasks)

Examples

Many tasks become easier if we have good feature

✦ Classification:

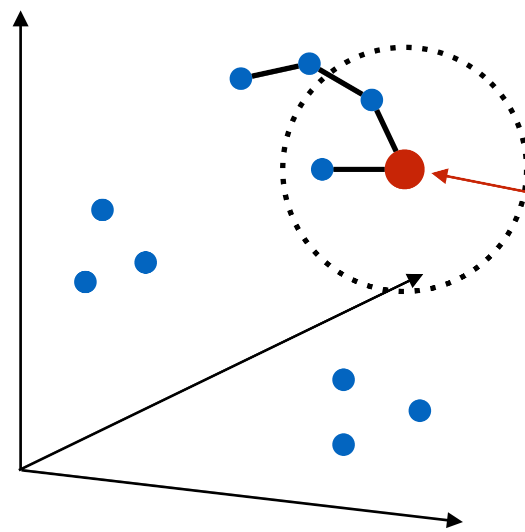
- SVM
- Logistic regression
- Nearest neighbor classifier
- Fine-tune the whole model (same or different data)



Examples

Many tasks become easier if we have good feature

- Classification:
  - SVM
  - Logistic regression
  - Nearest neighbor classifier
  - Fine-tune the whole model (same or different data)



- Visual search (retrieval):

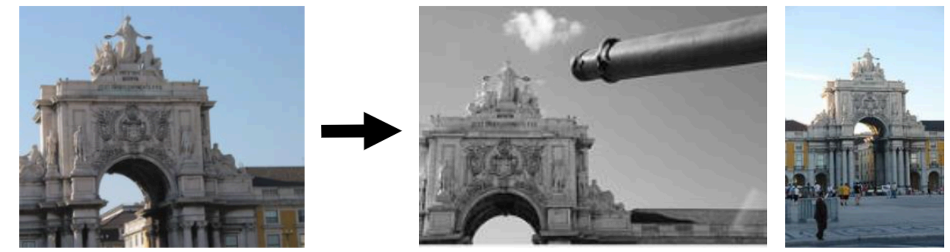
  - query with an image
  - retrieve similar objects or views
  - Euclidean nearest neighbor
  - NN graph distance

Examples

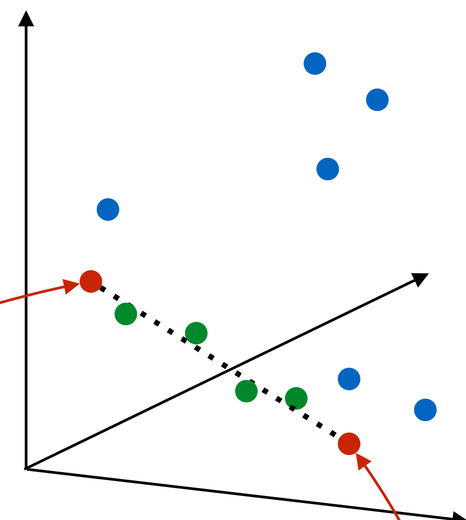
Many tasks become easier if we have good feature

- ✦ Classification:
  - SVM
  - Logistic regression
  - Nearest neighbor classifier
  - Fine-tune the whole model (same or different data)

- ✦ Visual search (retrieval):



  - query with an image
  - retrieve similar objects or views
  - Euclidean nearest neighbor
  - NN graph distance



- ✦ Data exploration

[Johnson et al. (2017)]

Word Vectors

Word Vectors

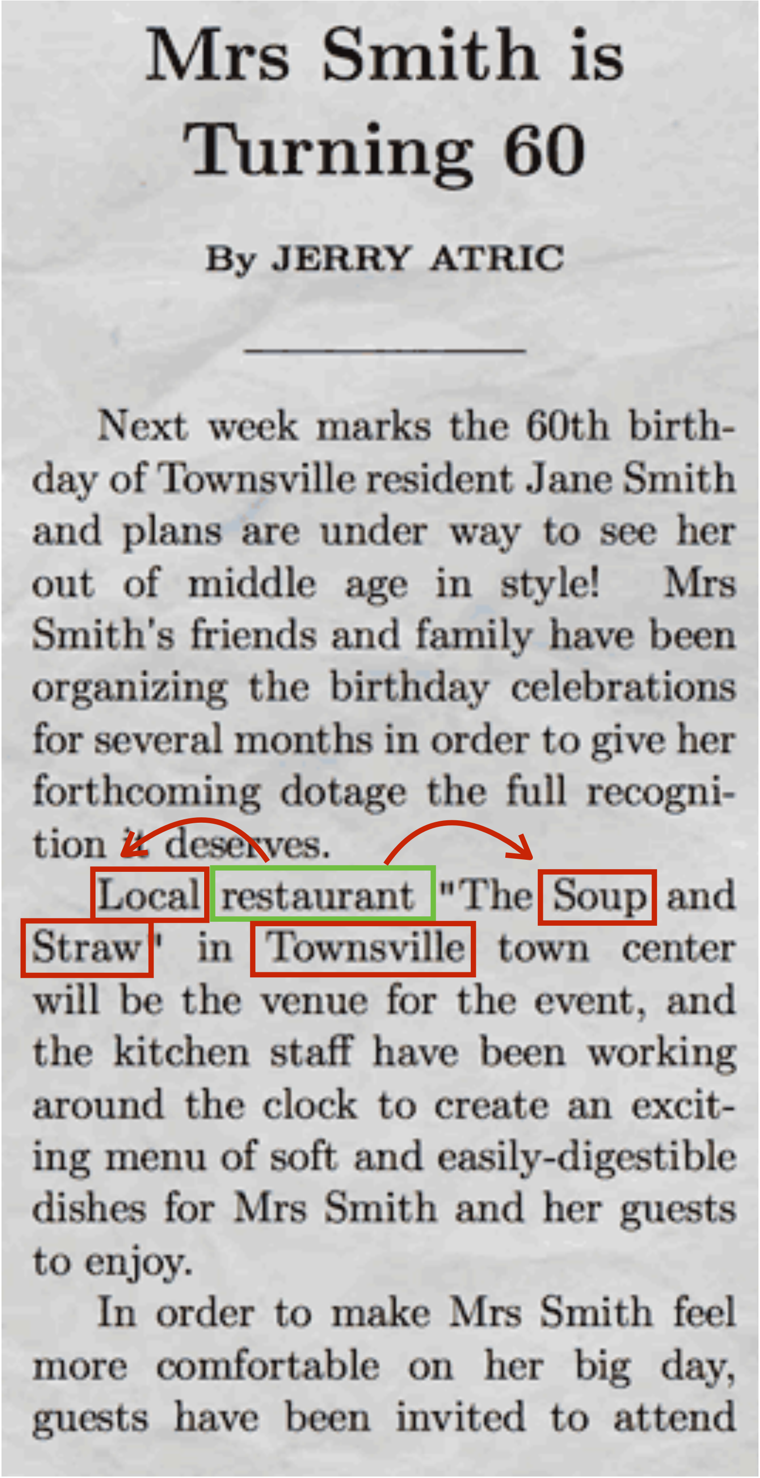
- Problem Formulation:
  - Assume a finite vocabulary I, $|I| = n$
  - Given a word x, predict nearby words y
- Simple Model:
  - Predict one word by categorical distribution $p(y|x)$
  - Let $v(x) \in \mathbb{R}^d$ – word vector for x
  - x is discrete and not structured — parameterize by a matrix V of all word vectors of size $n \times d$: $v(x) = V_{x,:}$
  - Define categorical distribution:

$$p(y|x) = \frac{e^{\langle u(y), v(x)\rangle}}{\sum_{y'} e^{\langle u(y'), v(x)\rangle}}$$

  - A different embedding for context words: $u(y) = U_{y,:}$
  - Learn via maximum likelihood classification:

$$\max_{U,V} \mathbb{E}_{t,t'} \Big[\log p(y_{t'}|x_t) \Big],$$

where t, t' – nearby positions in the text



Mrs Smith is Turning 60

By JERRY ATRIC

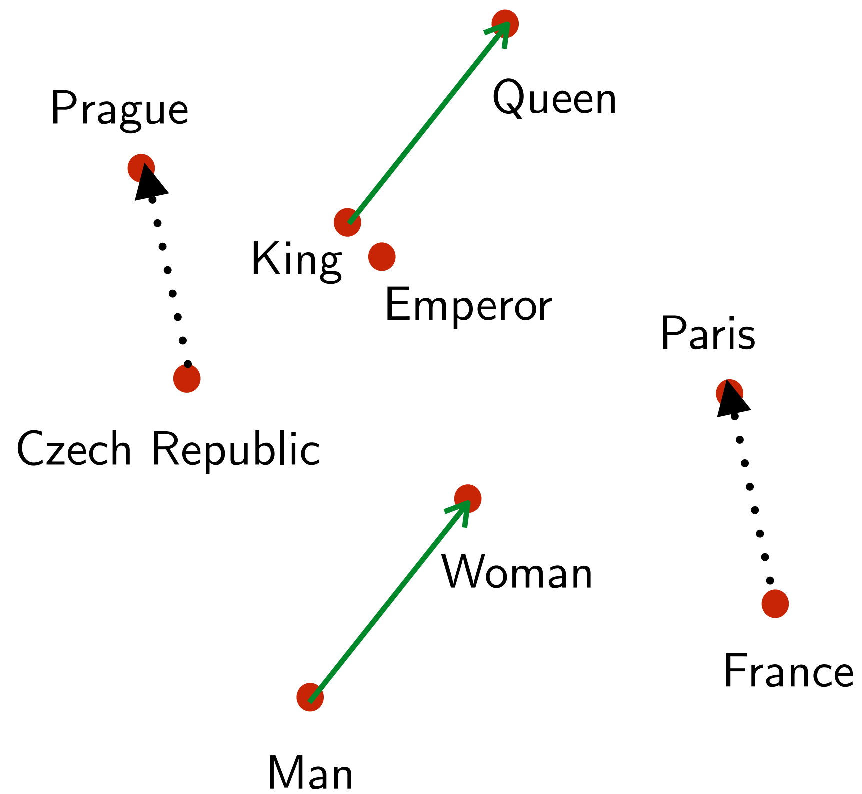
Next week marks the 60th birthday of Townsville resident Jane Smith and plans are under way to see her out of middle age in style! Mrs Smith's friends and family have been organizing the birthday celebrations for several months in order to give her forthcoming dotage the full recognition it deserves.

Local restaurant "The Soup and Straw" in Townsville town center will be the venue for the event, and the kitchen staff have been working around the clock to create an exciting menu of soft and easily-digestible dishes for Mrs Smith and her guests to enjoy.

In order to make Mrs Smith feel more comfortable on her big day, guests have been invited to attend

Word Vectors

- Learned a **representation** of each word x as the embedding $v(x) = V_{x,:} \in \mathbb{R}^d$



- Direction of $v(x)$ appears to capture abstract relations:
  - Semantic:

    "King" - "Man" + "Woman" $\approx$ "Queen"

    "Prague" - "Czech Republic" + "France" $\approx$ "Paris"

    "Czech" + "currency" $\approx$ "koruna"
  - Syntactic:

    "quick" - "quickly" $\approx$ "slow" - "slowly"
- Evaluated on a corpus of relation prediction tasks
- Trained without annotation, very useful representation for more complex task

Deep Similarity/Metric Learning

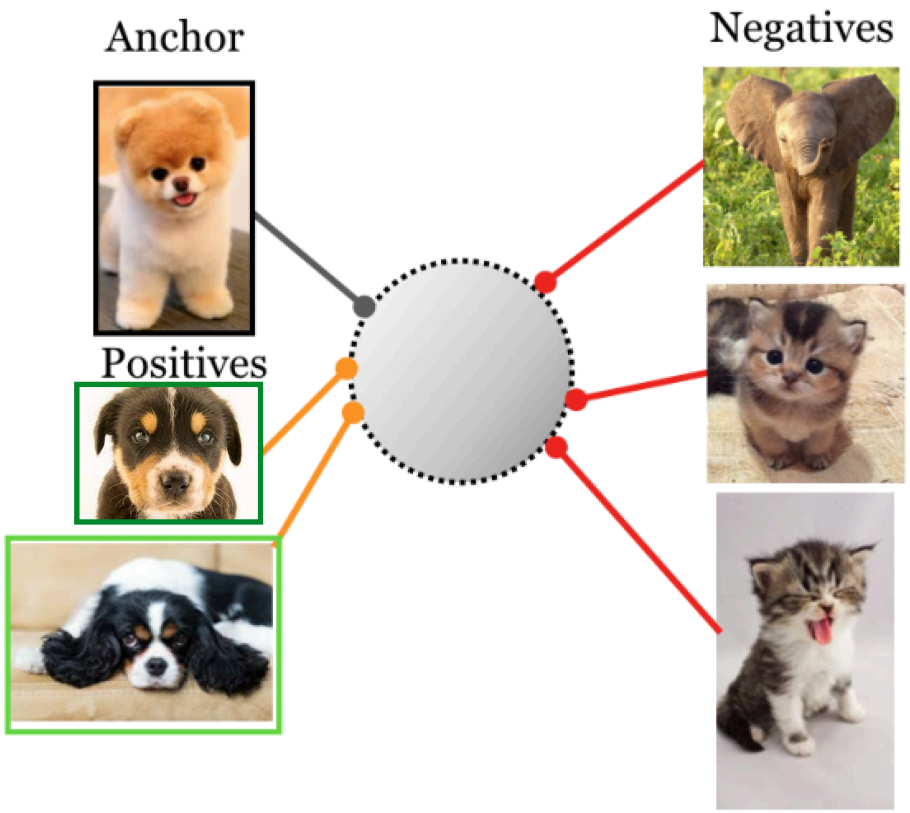
Similarity Learning

✦ Goals:

- learn the concept of **similarity** of two inputs
- quantify this similarity

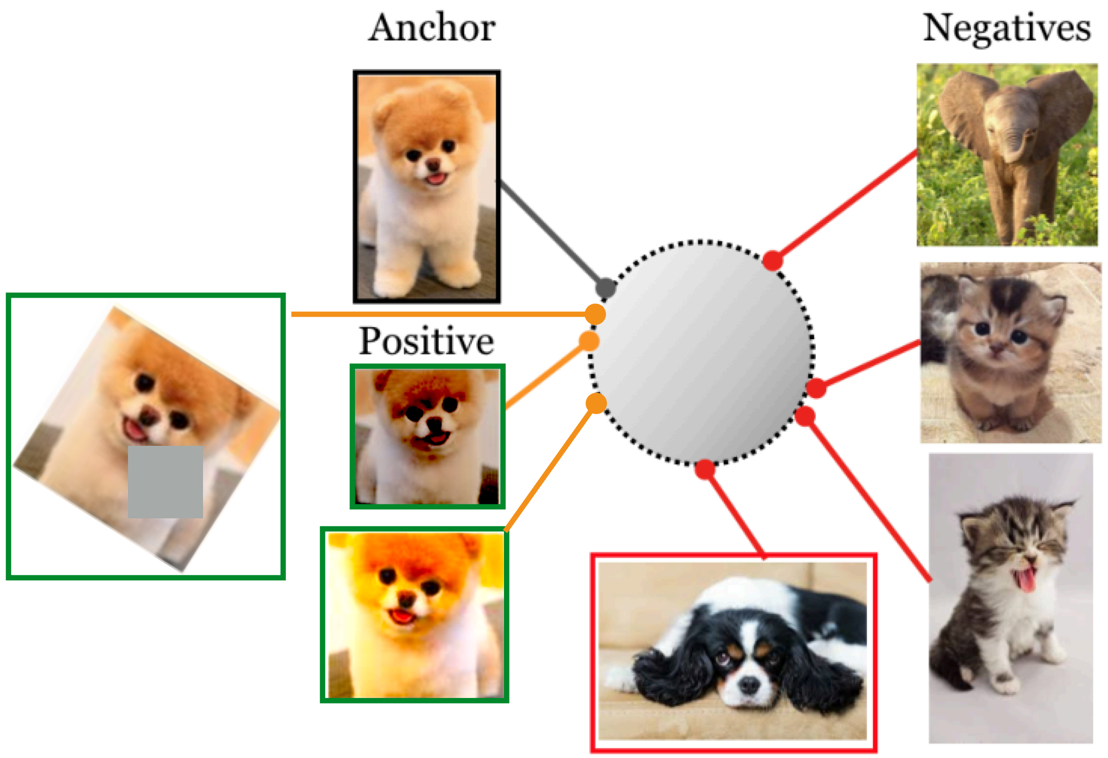
Supervised learning:

- Given examples of "similar" and "distinct" pairs e.g. by class label



Self-supervised learning:

- Generate "similar" pairs by identity-preserving transforms and random "distinct" pairs

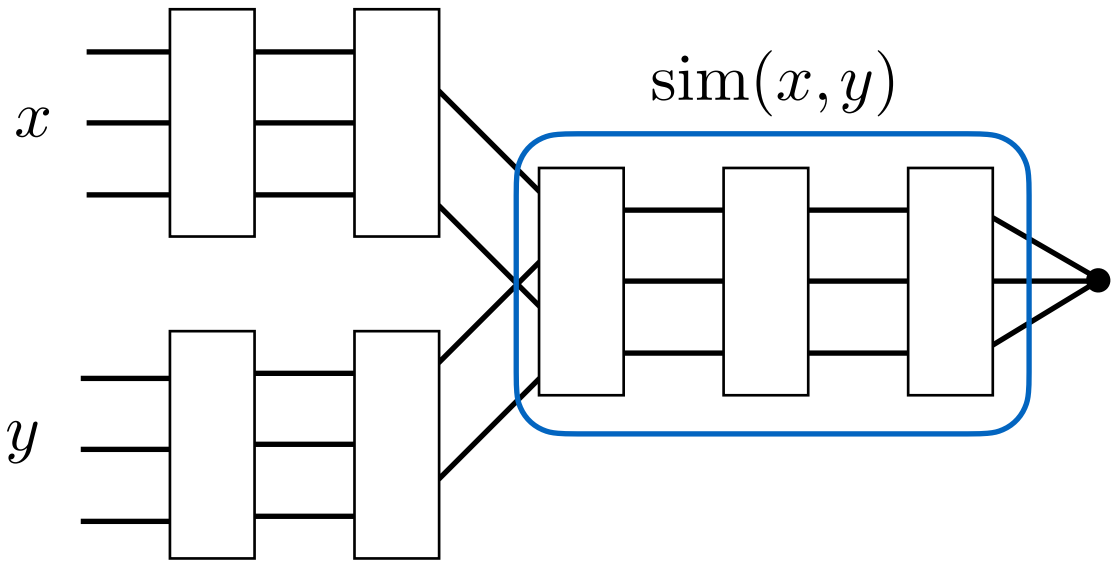


[Khosla et al. (2020) Supervised Contrastive Learning] original graphics edited for visualization

✦ "Similar pairs" must be closer in the feature space than "distinct" wrt learned **similarity**

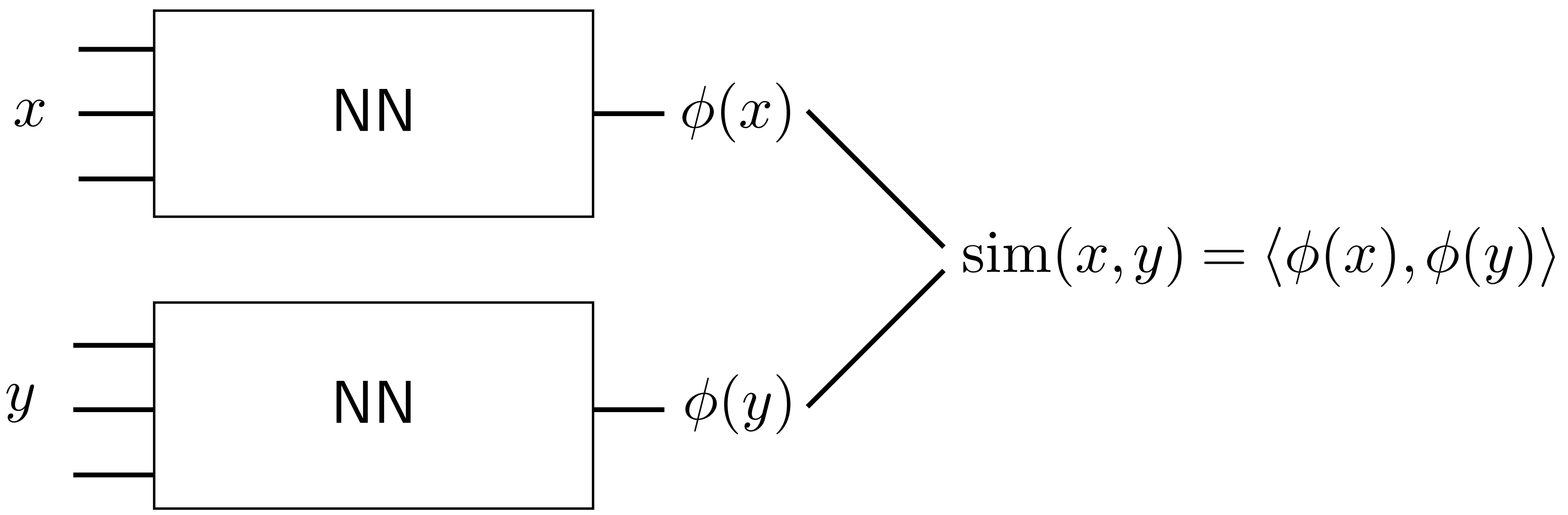
Model

✦ Approach 1: generic network with two inputs



first layers extract generic features -- can be shared

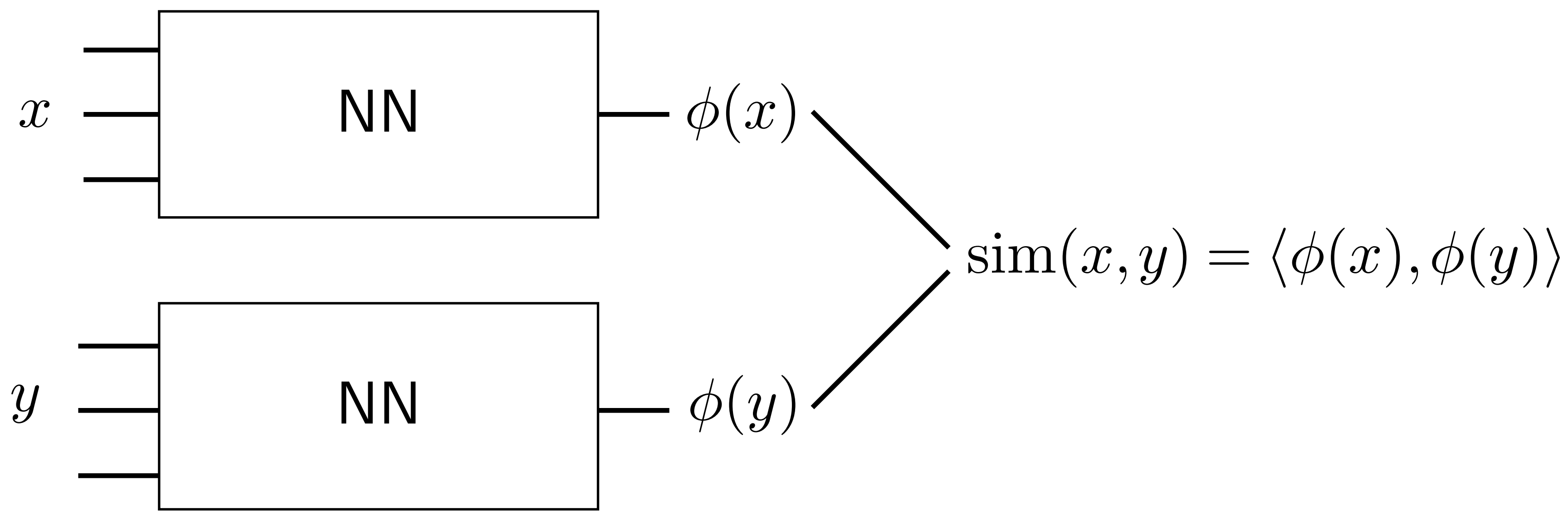
✦ Approach 2: network creates representations (embeddings / features)



- Inner product: $\langle \phi(x), \phi(y) \rangle$ can approximate any kernel $K(x, y)$

Similarity Function

- Network creates representations (embeddings / features)



- Inner product: $\mathrm{sim}(x, y) = \langle\phi(x), \phi(y)\rangle$
  Retrieval: Maximum Inner Product Search (MIPS) is more difficult (no triangle inequality)

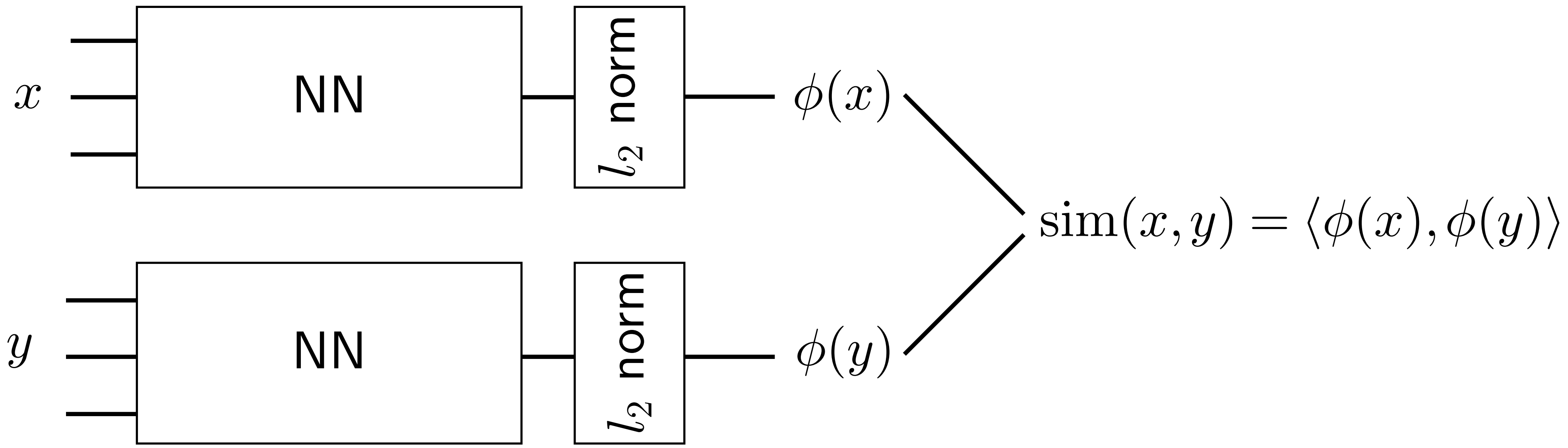
- Euclidean: $\mathrm{sim}(x, y) = -\|\phi(x) - \phi(y)\|^2$
  Retrieval: nearest neighbor search (NNS), sub-linear approximate methods

- Correlation: $\mathrm{sim}(x, y) = \frac{\langle\phi(x), \phi(y)\rangle}{\|\phi(x)\|\|\phi(y)\|}$
  Retrieval: correlation-NNS, sub-linear approximate methods

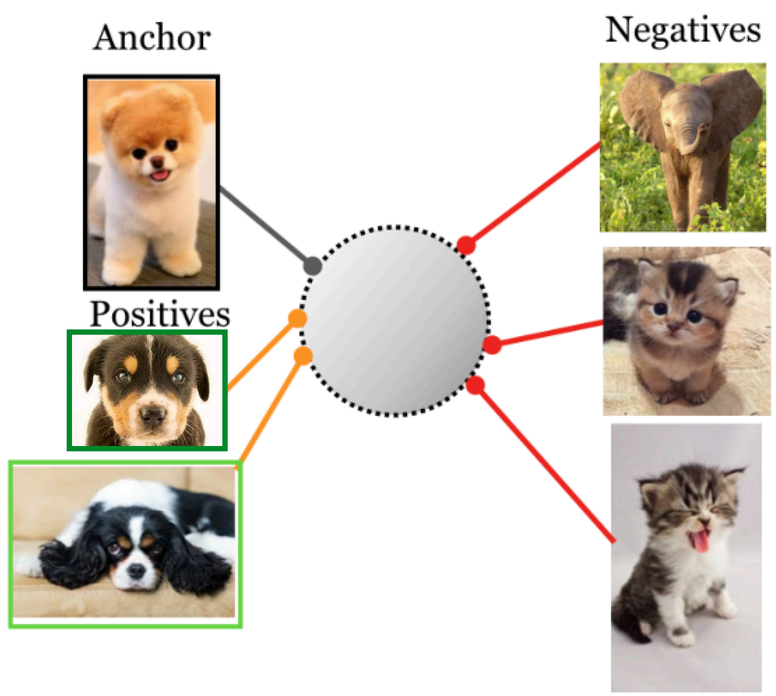
- All equivalent if $\|\phi(x)\| = 1$ for all x
- There are known mappings to approximate $\langle u, v\rangle$ with $\|P(u) - Q(v)\|^2$ or $\frac{\langle P(u), Q(v)\rangle}{\|P(u)\|\|Q(v)\|}$

Model

- Convenient common model: normalize representations (equivalent to using correlation for similarity)

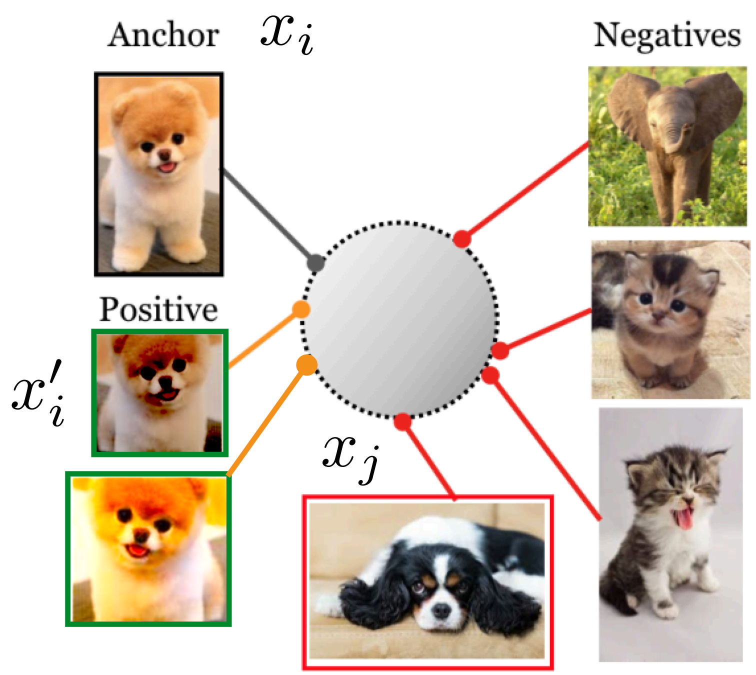


Representations live on a hypersphere



Instance Classification

- Training data: $x_1 \ldots x_N$
  - *Anchor*: x_i
  - *Positive*: $x' = T(x_i)$ – random transform
- Model as classification:
  - As many classes as there are instances (data points)
  - Score of instance i: $s_i = \phi(x_i)^\mathsf{T}\phi(x')$
  - $p(y{=}i|x') = \frac{e^{s_i}}{\sum_j e^{s_j}}$
  - Learning formulation: likelihood of classifying corectly
  - Large sum in the denominator $\rightarrow$ common solution is to restrict to a min-batch
- Properties:
  - Ensures instances can be discriminated
  - Enforces invariance to transformations



Self Supervised Contrastive

Dosovitskiy et al. (2014): Discriminative unsupervised feature learning with convolutional neural networks

Wu et al. (2018): Unsupervised Feature Learning via Non-Parametric Instance Discrimination]

Chen et al. (2020): A Simple Framework for **Contrastive Learning** of Visual Representations

Contrastive learning: "contrasting positive pairs against negative pairs"

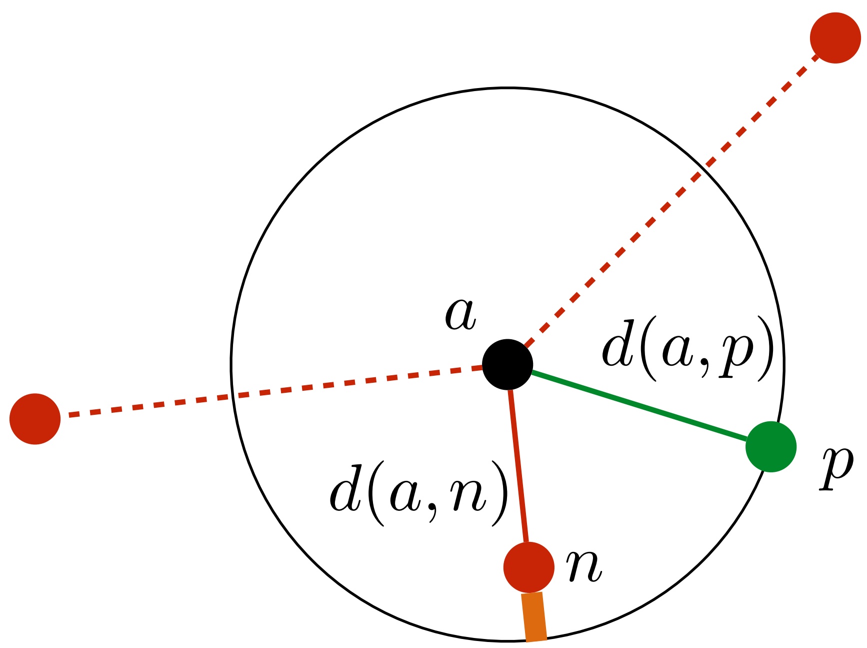
Triplet Losses

- Training data: $x_1 \dots x_N$
  - Positive pairs: $x_i \sim x_j$ for $(i,j) \in \mathcal{P}$
  - Negative pairs: $x_i \not\sim x_j$ for $(i,j) \in \mathcal{N}$
  - Example: known class label $\rightarrow$ similar if same class
- Desired separation property:
  - a – *anchor* (any data point)
  - p – *positive* sample for a
  - n – *negative*
  - let $D_p = \mathrm{d(a,p)}$ and $D_n = \mathrm{d(a,n)}$
  - Want:

$$D_p < D_n \qquad \forall p, n$$

$$D_p - D_n < 0$$

  - Hinge loss 1: $l = \sum_{n \in \mathcal{N}(a)} \max(0, D_p - D_n)$
  - Hinge loss 2: $l' = \max(0, \max_{n \in \mathcal{N}(a)} (D_p - D_n)) = \max_{x \in \mathcal{N}(a) \cup \{p\}} (D_p - D_x)$



need a negative violating the constraint the most --> **hard negative mining**

Smooth Triplet Loss = Log Likelihood Classification

- Hinge loss variant 1: $l = \sum_{n \in \mathcal{N}(a)} \max(0, D_p - D_n)$

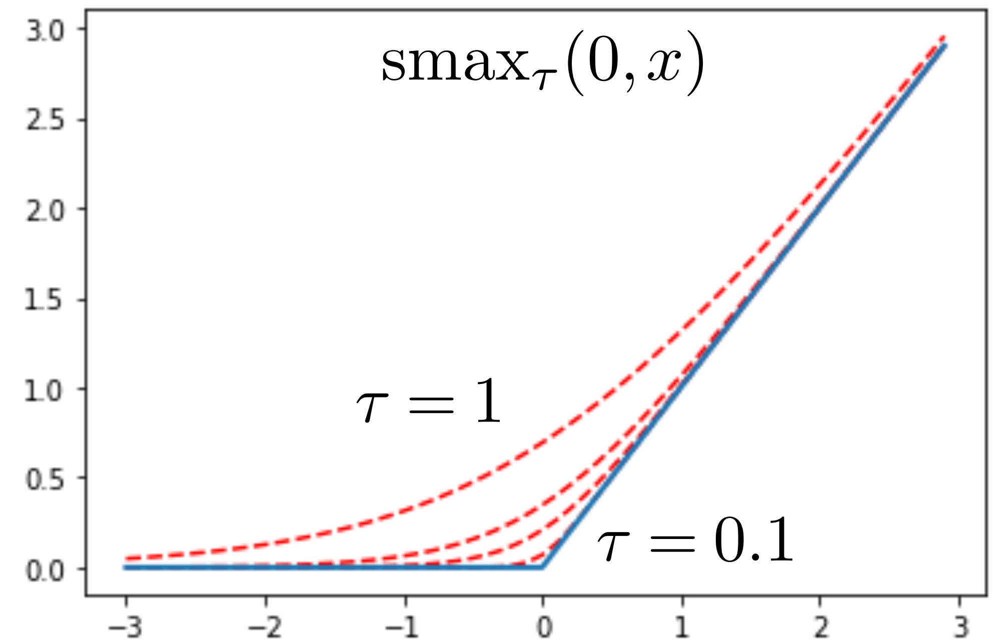
  Smooth Maximum:

  $\mathrm{smax}_\tau(x, y) = \tau \log(e^{x/\tau} + e^{y/\tau})$

  $Z \sim \mathrm{Logistic}(0, \tau)$

  $\mathbb{E}[\max(0, x + Z)] = \mathrm{smax}_\tau(0, x) = \tau \log(1 + e^{x/\tau})$

  Smooth hinge loss 1: $l = \sum_{n \in \mathcal{N}(a)} \log(1 + e^{D_p - D_n})$



  NLL of logistic regression, all positive vs negative pairs, used in e.g. [Sohn 2016]

- Hinge loss variant 2: $l' = \max_{x \in \mathcal{N}(a) \cup \{p\}} (D_p - D_x)$

$$\mathrm{smax}_{x \in \mathcal{N}(a) \cup \{p\}} (D_p - D_x) = \log(1 + \sum_{x \in \mathcal{N}(a)} e^{D_p - D_x}) = \log(1 + e^{D_p} \sum_{x \in \mathcal{N}(a)} e^{-D_x})$$

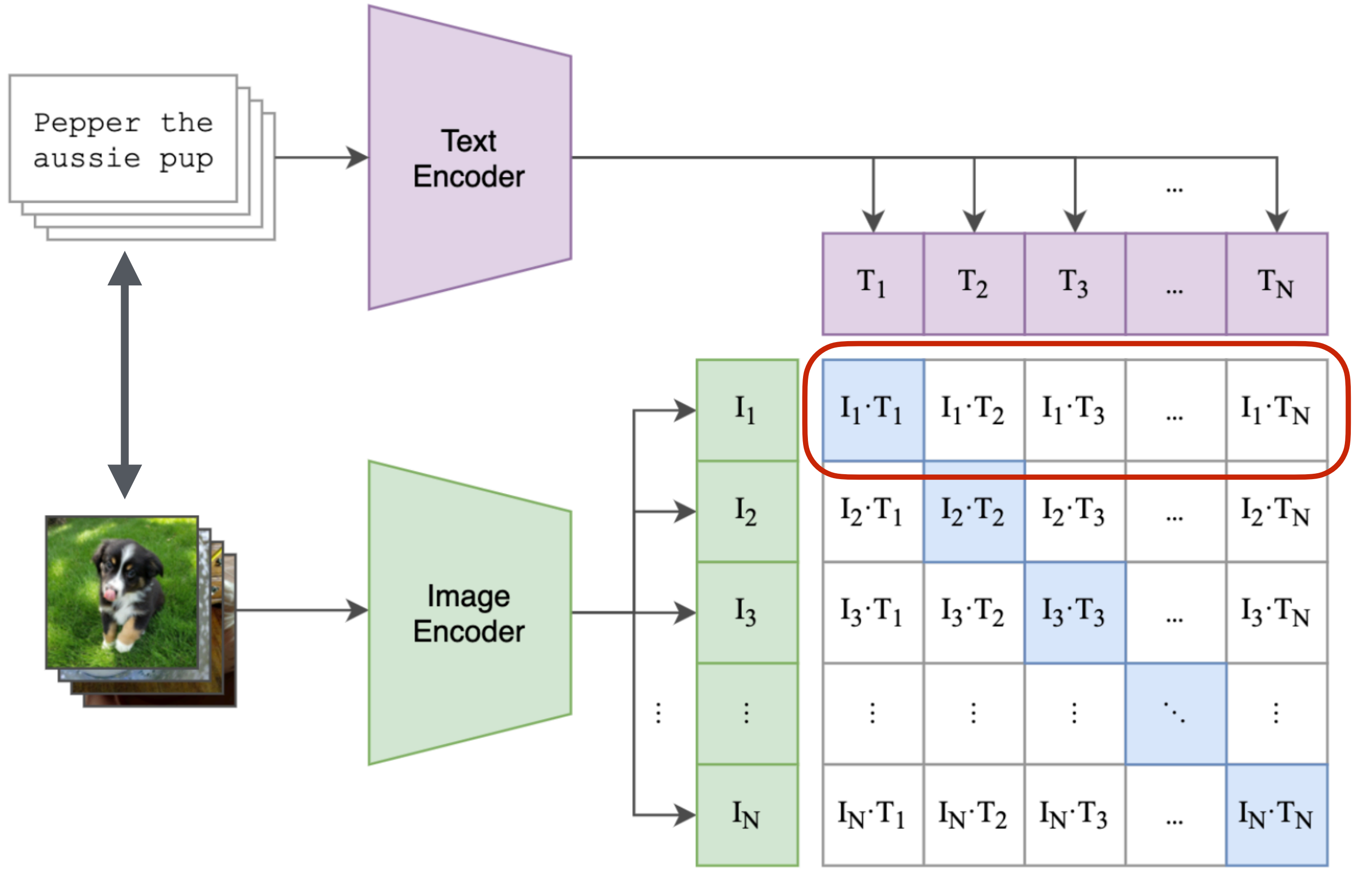
$$= -\log \frac{1}{1 + e^{D_p} \sum_{x \in \mathcal{N}(a)} e^{-D_x}} = -\log \frac{e^{-D_p}}{\sum_{x \in \mathcal{N}(a) \cup \{p\}} e^{-D_x}}$$

  NLL of softmax classification, predicting the positive out of all candidates

  There is a noisy max interpretation as well

Cross-Modality Representations

CLIP: Connecting Text and Images



- In each batch of image-text pairs with embeddings I_i, T_j:
  - Predict which text corresponds to image i, model: $p_1(j|i) = \frac{e^{\langle I_i, T_j \rangle / \tau}}{\sum_{j'} e^{\langle I_i, T_{j'} \rangle / \tau}}$
  - Predict which image corresponds to text j, model: $p_2(i|j) = \frac{e^{\langle I_i, T_j \rangle / \tau}}{\sum_{i'} e^{\langle I_{i'}, T_j \rangle / \tau}}$
  - Learning: symmetric cross-entropy loss:

$$-\sum_i \Big(\log p_1(j|i) + \log p_2(i|j) \Big)$$

[Radford et al. 2021: Learning Transferable Visual Models From Natural Language Supervision]

Summary

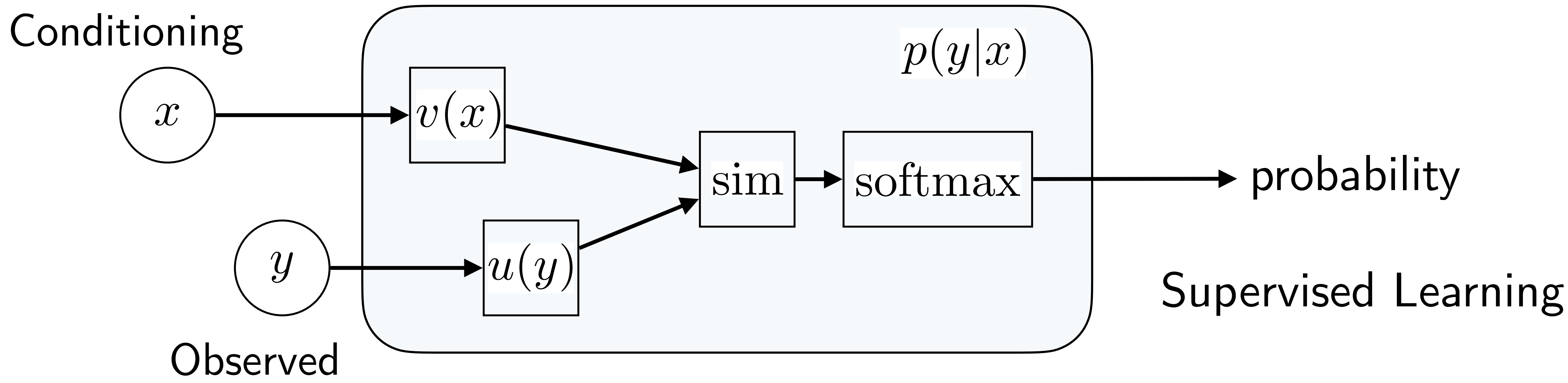
- Word Vectors:
  - Non-parametric features, similarity $\text{sim}(x, y) = \langle U_y, V_x \rangle$
  - Softmax classification = smooth triplet loss
- Contrastive Learning methods:
  - Deep normalized features, similarity $\text{sim}(x, y) = \langle \phi(x), \phi(y) \rangle$
  - Triplet loss variants (hard /smooth)
- Instance Classification:
  - Softmax classification = smooth triplet loss
- CLIP:
  - Two encoders: for images $I(y)$ and text $T(x)$, similarity $\text{sim}(x, y) = \langle I(x), T(y) \rangle$
  - Softmax classification with anchors of both kinds
- Family of triplet losses, smoothness controlled by hyperparameter τ

Probabilistic Latent Representations

Latent Variable Models

- Word Vectors

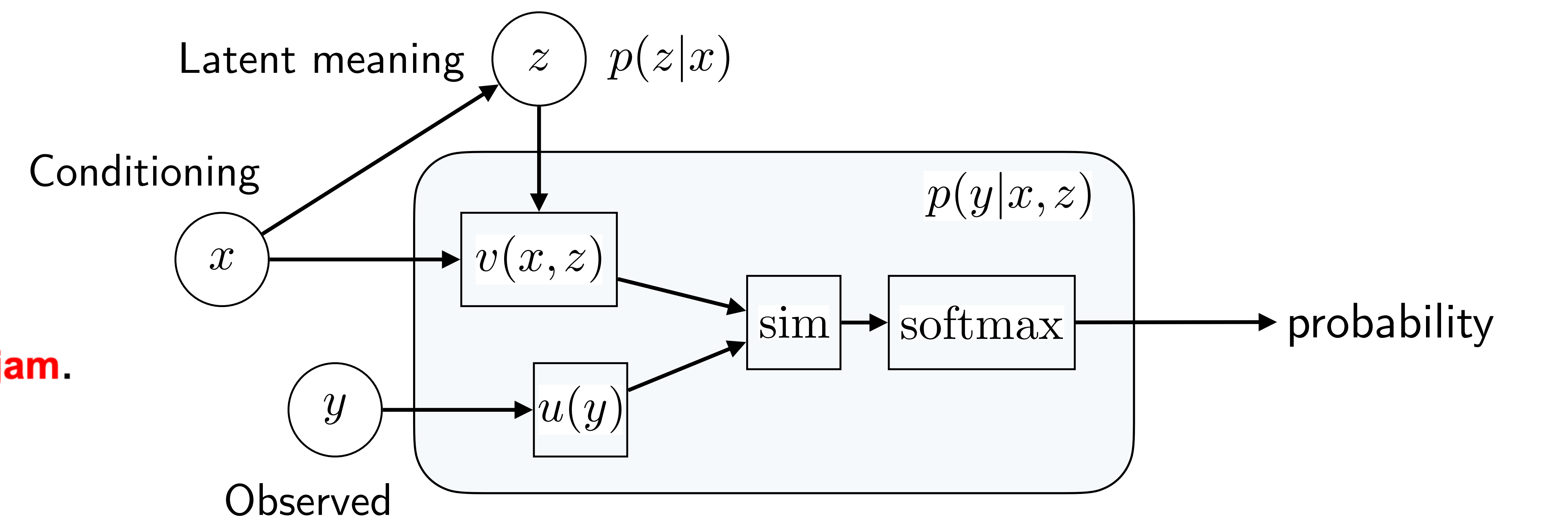
Be careful not to **jam** your finger in the door.
y x



- Multi-sense Word Vectors

I eat grape **jam**.

I was in a traffic **jam**.



Semi-Supervised / Unsupervised Learning

Multi-sense Word Vectors

Be careful not to **jam** your finger in the door.

Observation y

Conditioning word, x

Latent meaning $z \in \{1, \ldots 5\}$

Word embedding $u(y)$, word embedding $v(x, z)$

- Model:

  $p_\theta(y|x, z)$ – model of observations knowing the latent state $\quad \propto \exp(\mathrm{sim}(u(x), v(z, c)))$

  $p_\theta(z|x)$ – model of latent states $\quad$ table

  Conditional generative model: $p_\theta(y, z|x) = p_\theta(y|z, x) p_\theta(z|x)$

- Maximum likelihood learning:

  Observed: conditioning word x, neighboring word y

  Marginal likelihood $p_\theta(y|x) = \sum_z p_\theta(y, z|x) = \sum_z p_\theta(y|x, z) p_\theta(z|x)$

  NLL: $l(\theta) = \sum_i \log \sum_z p_\theta(y_i|z, x_i) p_\theta(z|x_i)$

Multi-Sense Word Vectors

- Learned prior distribution

WORD	$p(z\|x)$	NEAREST NEIGHBOURS
python	0.33	monty, spamalot, cantsin
	0.42	perl, php, java, c++
	0.25	molurus, pythons
apple	0.34	almond, cherry, plum
	0.66	macintosh, iifx, iigs

- Inference $p(z|x, y)$

`Our train has departed from Waterloo at 1100pm`

Probabilities of meanings
0.948032
0.00427984
0.000470485
0.0422029
0.0050148

Closest word:

"paddington"
"euston"
"victoria"
"liverpool"
"moorgate"
"via"
"london"

`Who won the Battle of Waterloo?`

Probabilities of meanings
0.0000098
0.997716
0.0000309
0.00207717
0.00016605

"sheriffmuir"
"agincourt"
"austerlitz"
"jena-auerstedt"
"malplaquet"
"königgrätz"
"mollwitz"
"albuera"