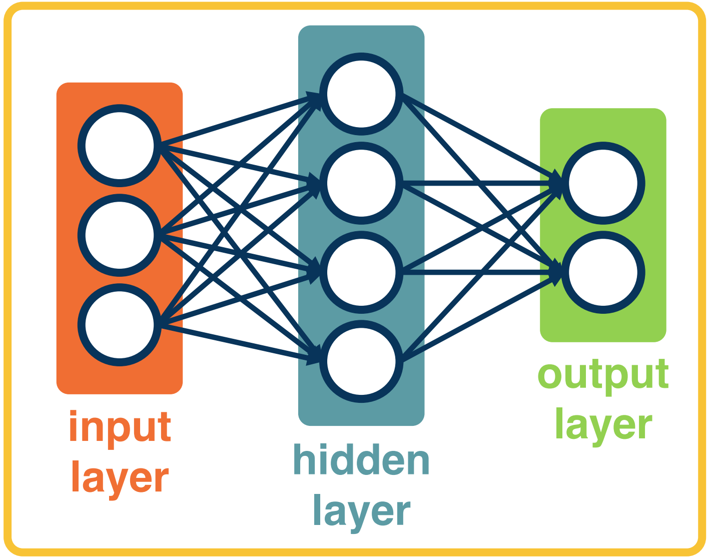
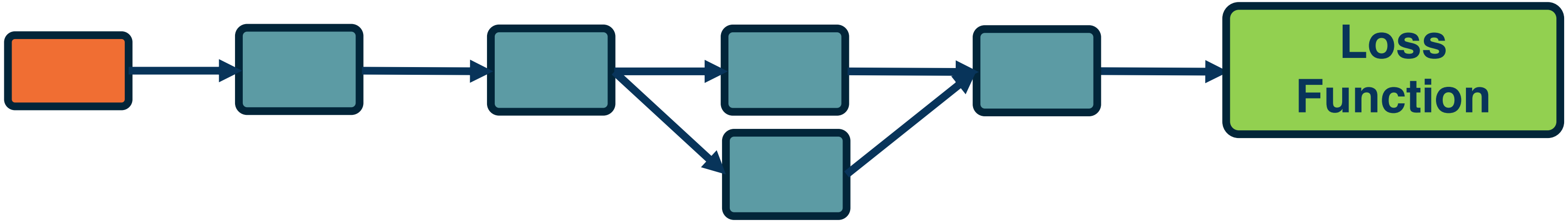
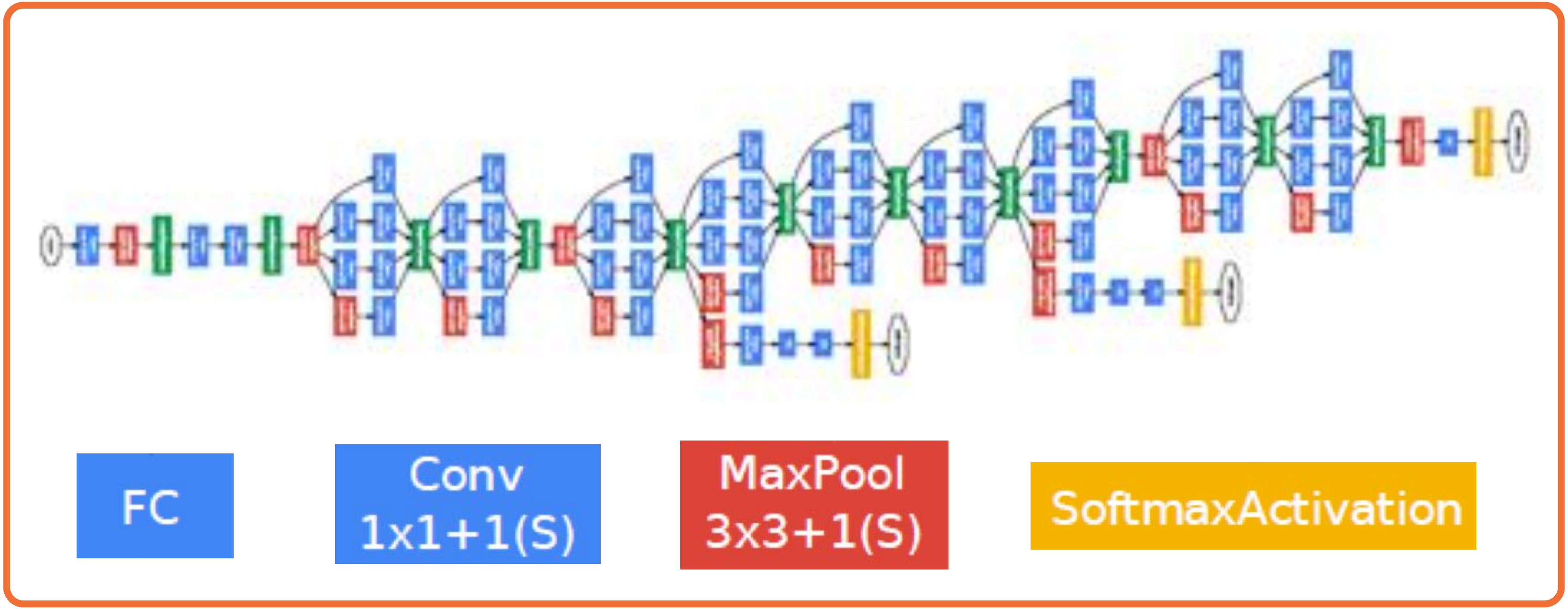
Structures and Structured Representations

We have seen how **to build and optimize deep feedforward architectures** consisting of linear & non-linear (e.g. ReLU) layers

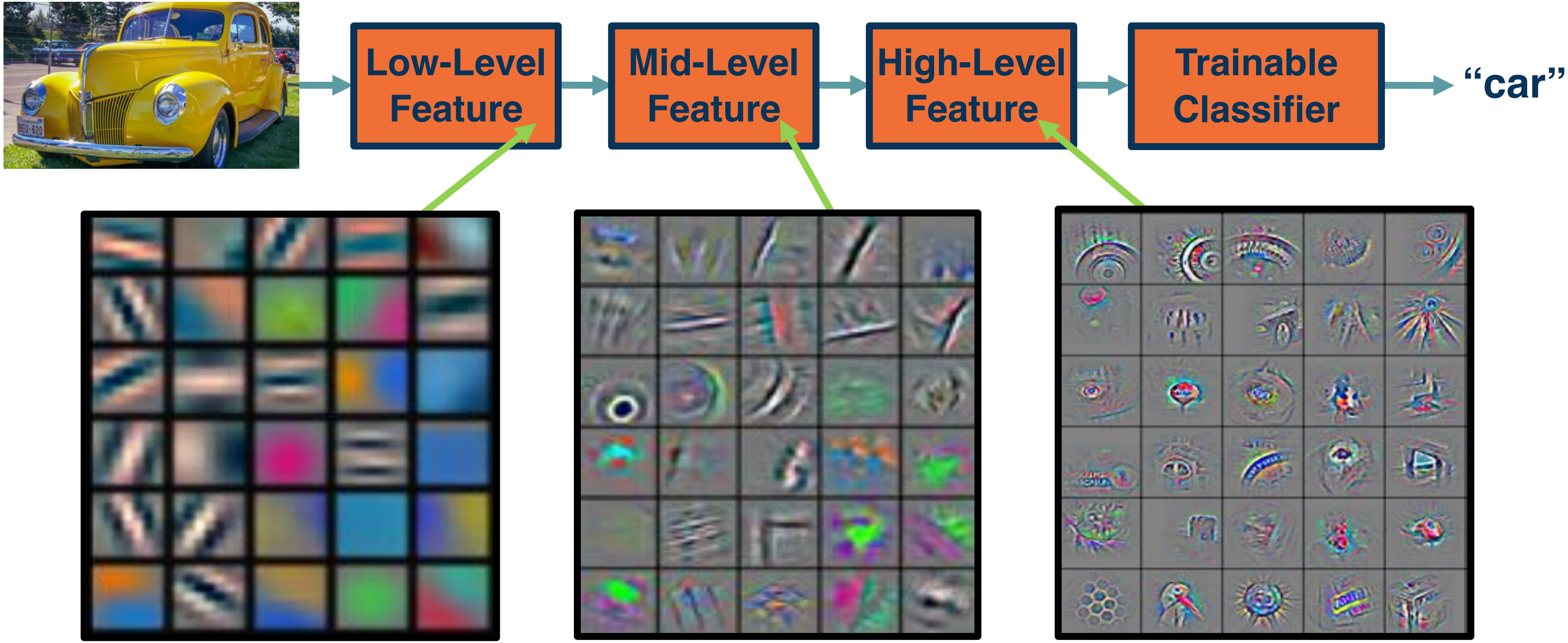
- This can be generalized to **arbitrary computation graphs**
- **Backpropagation and automatic differentiation** can be used to optimize all parameters via **gradient descent**







From: Szegedy et al. Going deeper with convolutions



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Adapted from slides by: Marc'Aurelio Ranzato, Yann LeCun

Deep Learning = Hierarchical Compositionality

VISION

pixels → edge → texton → motif → part → object

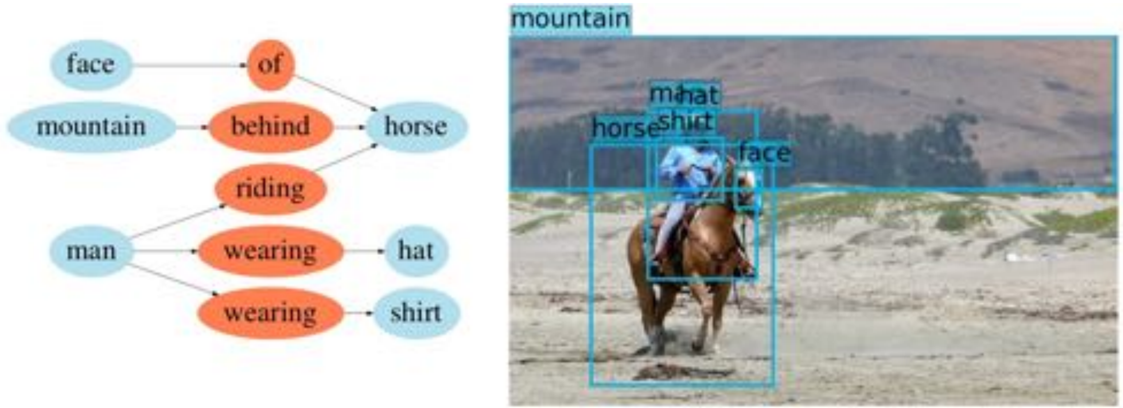
SPEECH

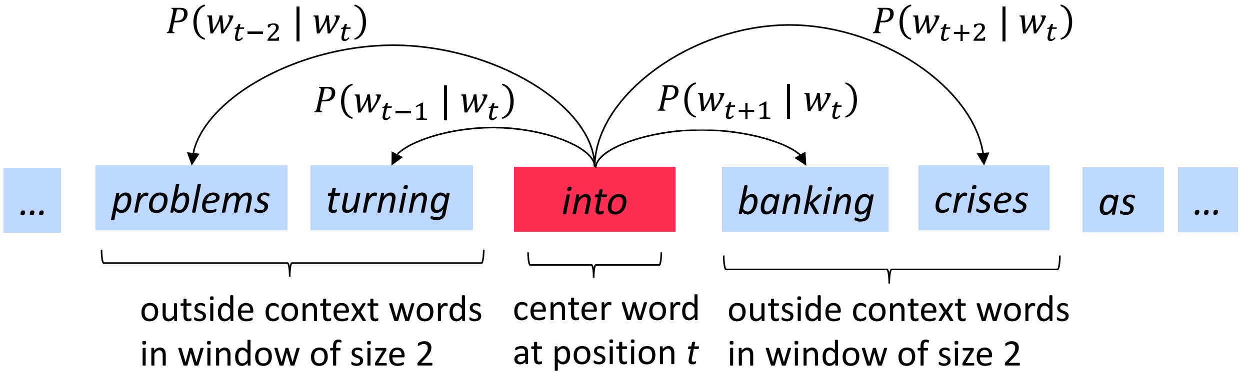
sample → spectral band → formant → motif → phone → word

NLP

character → word → NP/VP/.. → clause → sentence → story

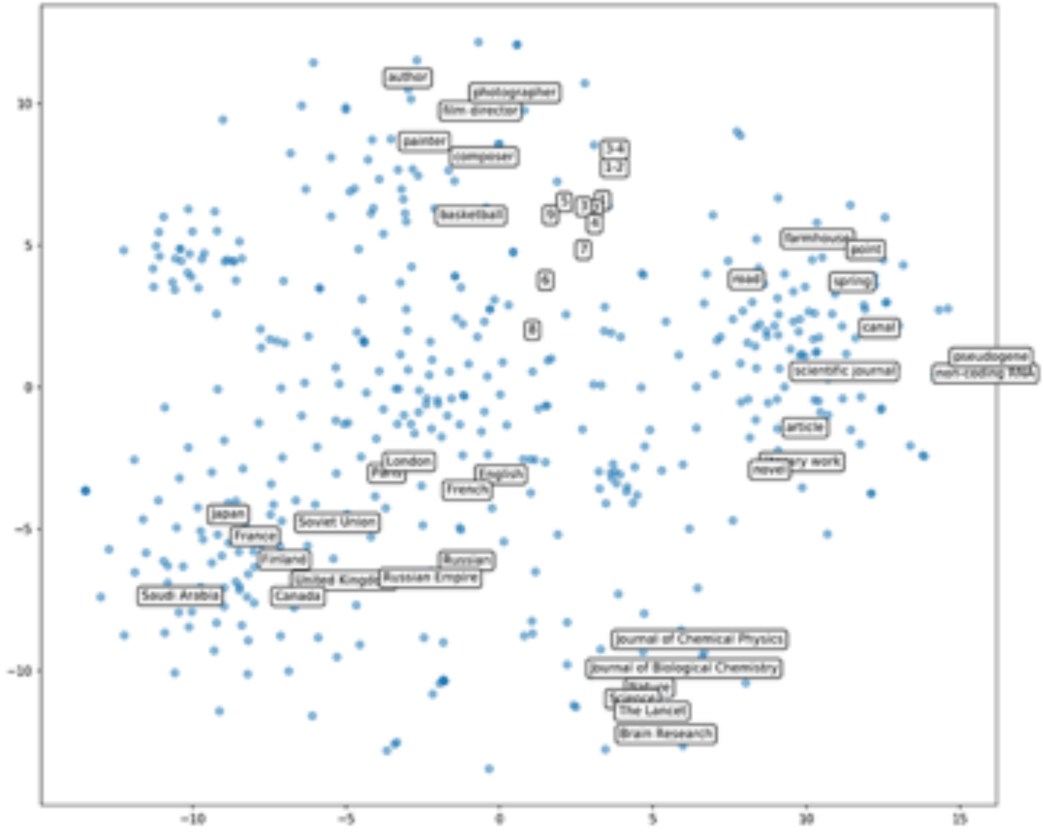
Adapted from slides by: Marc'Aurelio Ranzato, Yann LeCun





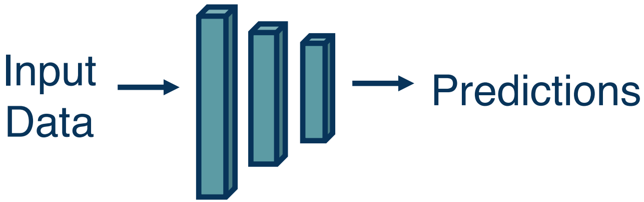
Xu et al., "Scene Graph Generation by Iterative Message Passing", 2017

Relationships are Important

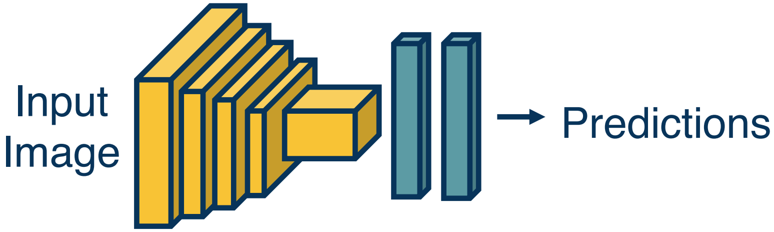


Embedding Wikidata Graph [Lerer et al. 19']

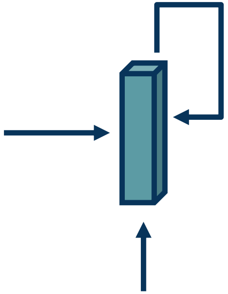
https://github.com/facebookresearch/PyTorch-BigGraph



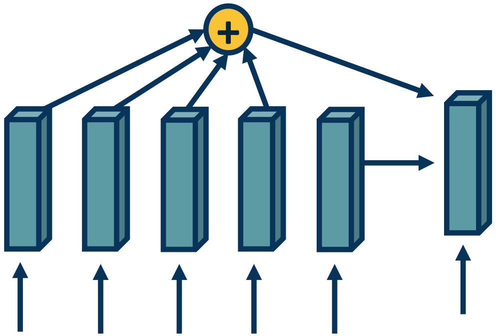
Fully Connected Neural Networks



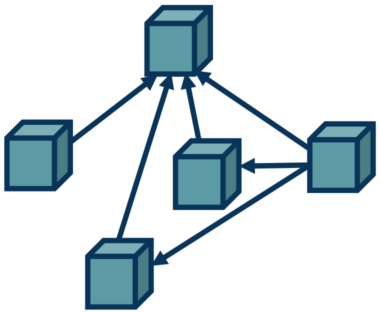
Convolutional Neural Networks



Recurrent Neural Networks

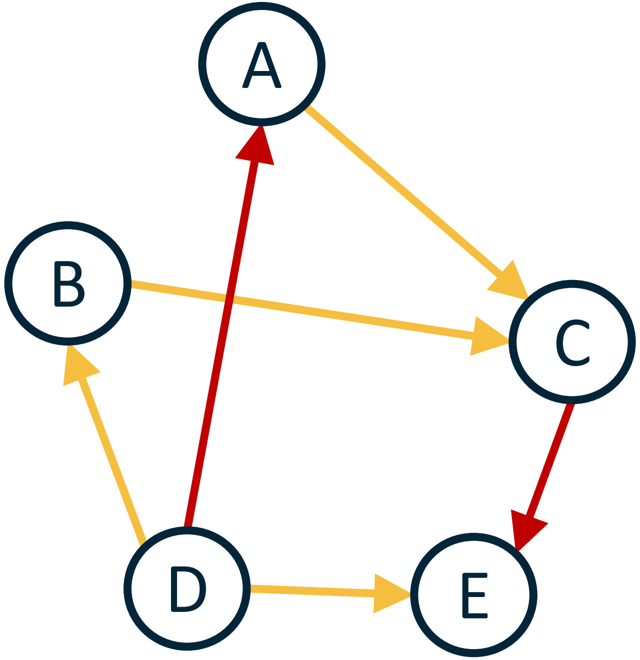


Attention-Based Networks



Graph-Based Networks

The Space of Architectures



A Multi-Relation Graph

Embedding: A learned map from entities to vectors of numbers that encodes similarity

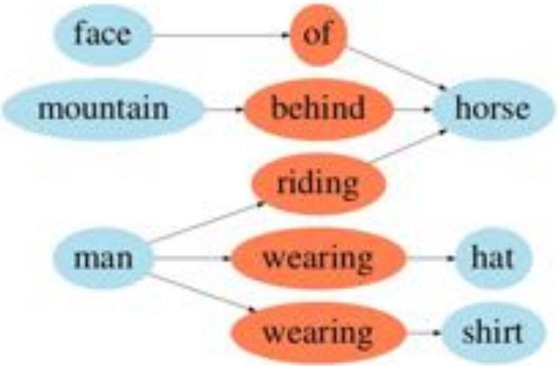
- Word embeddings: word → vector
- Graph embeddings: node → vector

Graph Embedding: Optimize the objective that **connected nodes have more similar embeddings** than unconnected nodes via gradient descent.

Slide Credit: Adam Lerer

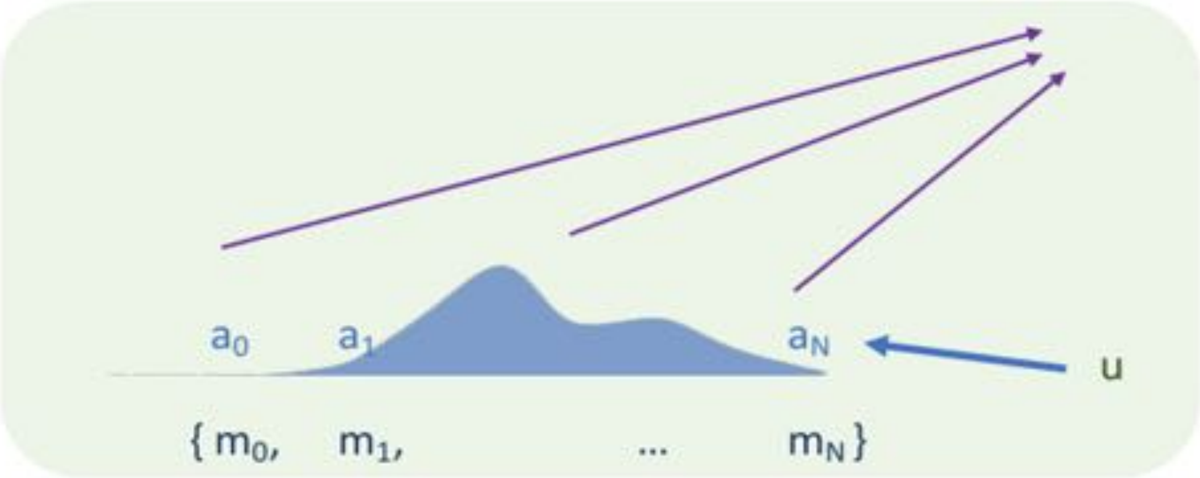
When representing structured information, several things are important:

- **State:** Compactly representing all the data we have processed thus far
- **"Neighborhoods"**: What other elements to incorporate?
  - Can be seen as selecting from a set of elements
  - Typically done with some similarity measure or attention
- **Propagation of information**: How to update information given selected elements



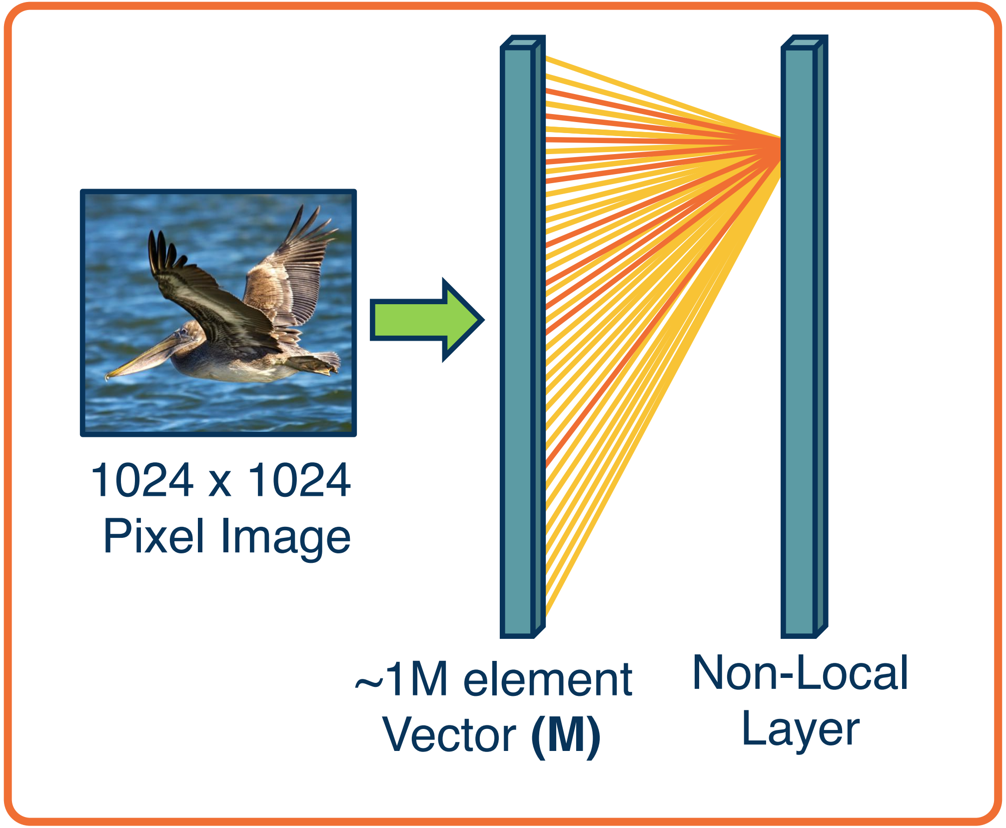
Slide Credit: Adam Lerer

- Given a set of vectors $\{\boldsymbol{u_1}, \dots, \boldsymbol{u_N}\}$ and a "query" vector $\boldsymbol{q}$
- We can select the most similar vector to $\boldsymbol{q}$ via $\boldsymbol{p} = \boldsymbol{Softmax}(\boldsymbol{Uq})$



$$a_j = \frac{e^{u_j \cdot q}}{\sum_k e^{u_k \cdot q}}$$

$$\textbf{output} = \sum_k a_k u_k$$



$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j)$$

- Output pixel i, and j representing all possible input positions
- f is similarity function, g is computes representation of input element j
- **Examples:**

$$f(x_i, x_j) = e^{x_i^T x_j}$$

$$g(x_j) = W_g x_j$$

Wang et al., "Non-local Neural Networks", 2017

Example: Non-Local Neural Networks

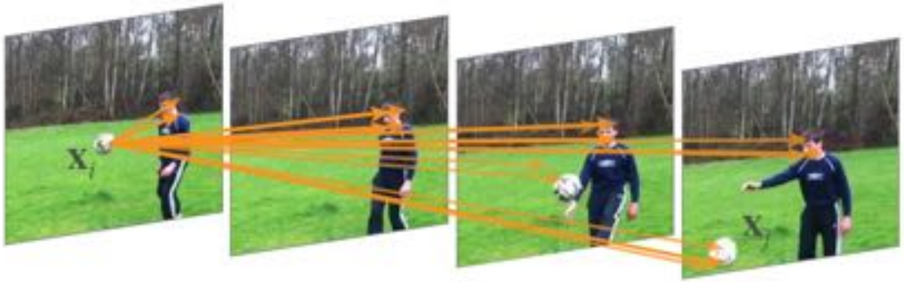


Figure 1. A spacetime ***non-local*** operation in our network trained for video classification in Kinetics. A position $\mathbf{x}_i$'s response is computed by the weighted average of the features of *all* positions $\mathbf{x}_j$ (only the highest weighted ones are shown here). In this example computed by our model, note how it relates the ball in the first frame to the ball in the last two frames. More examples are in Figure 3.

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j)$$

- Output pixel *i*, and *j* representing all possible input positions
- *f* is similarity function, *g* is computes representation of input element *j*
- **Examples:**

$$f(x_i, x_j) = e^{x_i^T x_j}$$

$$g(x_j) = W_g x_j$$

Wang et al., "Non-local Neural Networks", 2017

Example: Non-Local Neural Networks