CS 639: Foundation Models
Self—Supervised Learning

Fred Sala

University of Wisconsin-Madison

Feb. 3, 2026

Announcements

- Midterm: **Weds. March 11th**
    - HW 1: Coming out **Thursday**
- **Resources**
    - https://www.deeplearningbook.org/ : Solid intro to DL
- Class roadmap:

Tuesday Feb. 3	Self-Supervised Learning
Thursday Feb. 5	Guest Lecture
Tuesday Feb. 10	Transformers and Attention I
Thursday Feb. 12	Transformers and Attention II

Start FMs and Arch

Outline

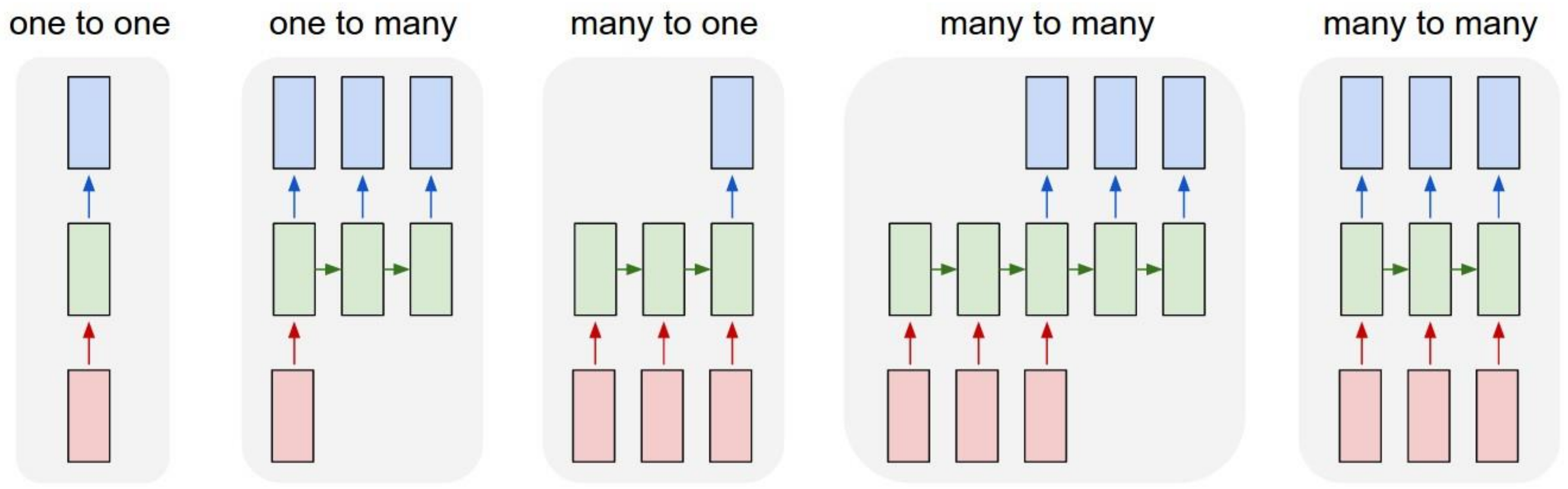
- **Sequence Models & Graph Models**
  - Recurrent neural networks, architecture, LSTMs, graphs and graph-based models
- **Self-Supervised Learning**
  - Motivation, basic idea, masking, autoencoders, student-teacher methods
- **Contrastive Learning**
  - Intuition, losses, getting positives and negatives, frameworks

Outline

- **Sequence Models & Graph Models**
  - Recurrent neural networks, architecture, LSTMs, graphs and graph-based models
- **Self-Supervised Learning**
  - Motivation, basic idea, masking, autoencoders, student-teacher methods
- **Contrastive Learning**
  - Intuition, losses, getting positives and negatives, frameworks

Break & Questions

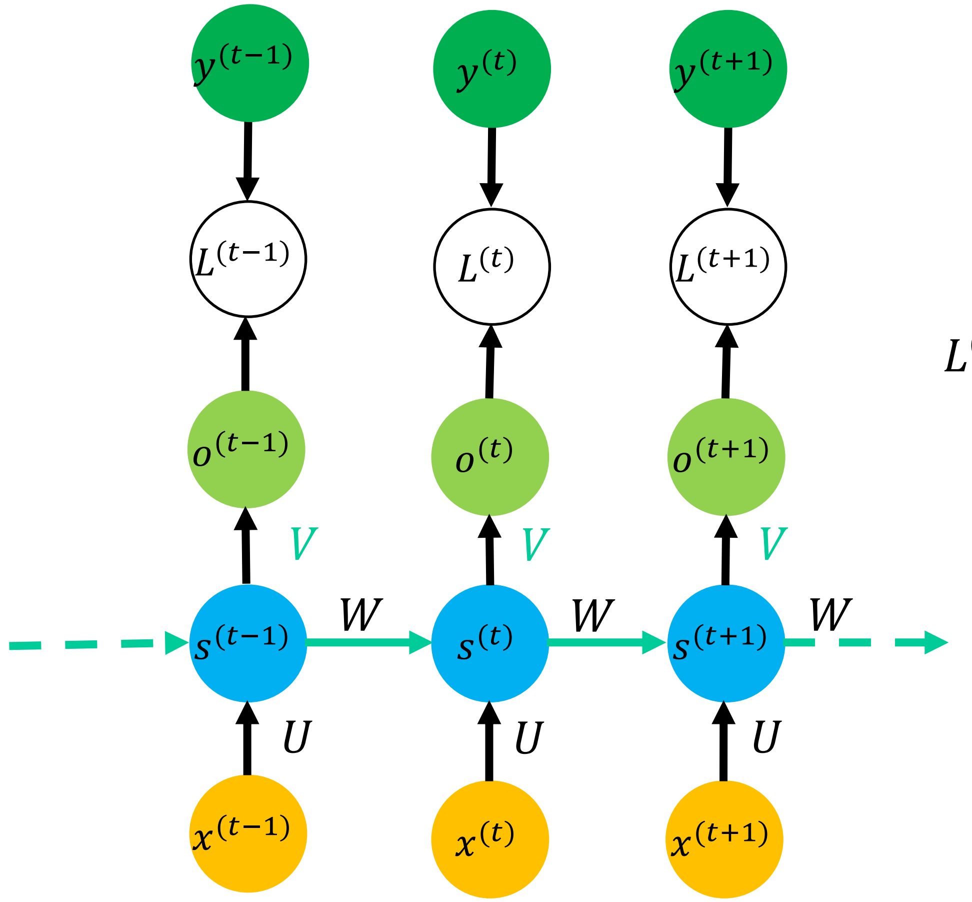
Tasks We Can Handle with NNs?



- Mostly talked about (1) so far
  - Others: need a new kind of model

Neural Networks: Simple RNNs

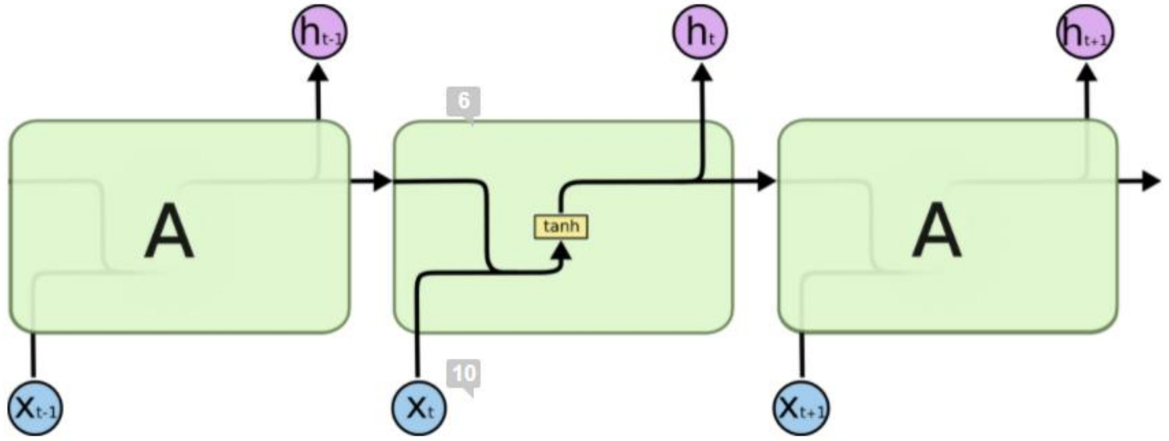
- Classical RNN variant:



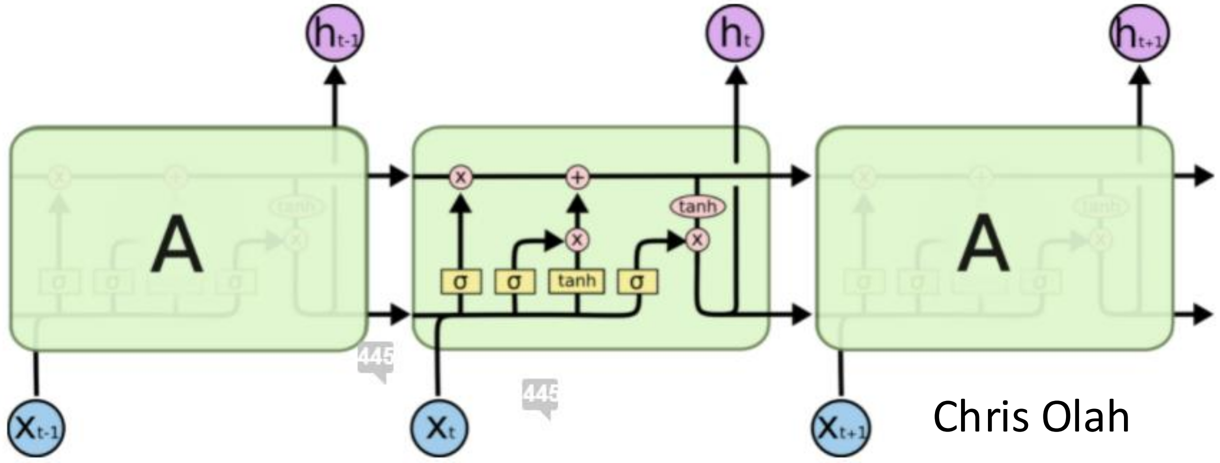
$$a^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$$
$$s^{(t)} = \tanh\left(a^{(t)}\right)$$
$$o^{(t)} = c + Vs^{(t)}$$
$$\hat{y}^{(t)} = \mathrm{softmax}\left(o^{(t)}\right)$$
$$L^{(t)} = \mathrm{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})$$

Neural Networks: LSTMs

- RNN: can write structure as:



- Long Short-Term Memory: deals with problem. Cell:

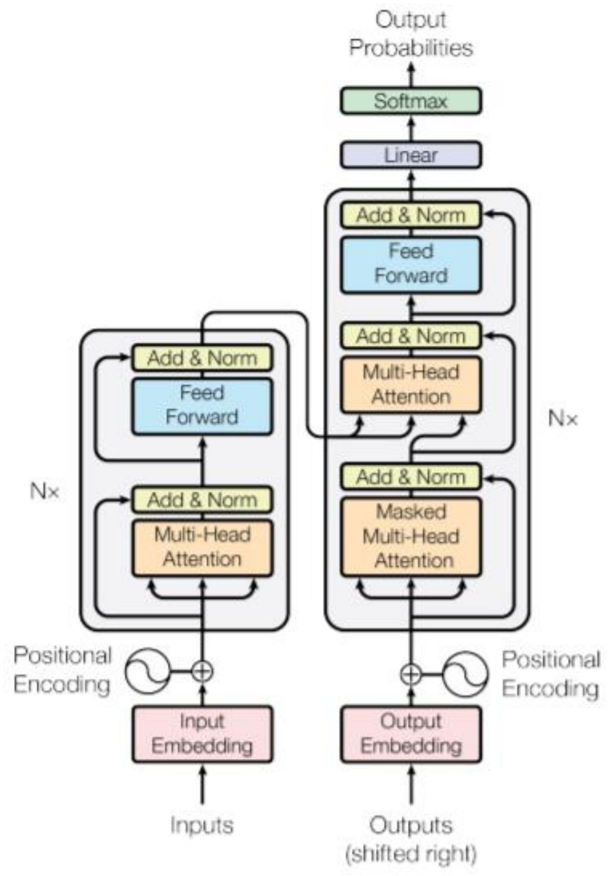


Chris Olah

Neural Networks: Transformers

- Initial goal for an architecture: **encoder-decoder**
    - Get **rid of recurrence**
    - Replace with **self-attention**

- Architecture
    - The famous picture you've seen
    - Centered on self-attention blocks

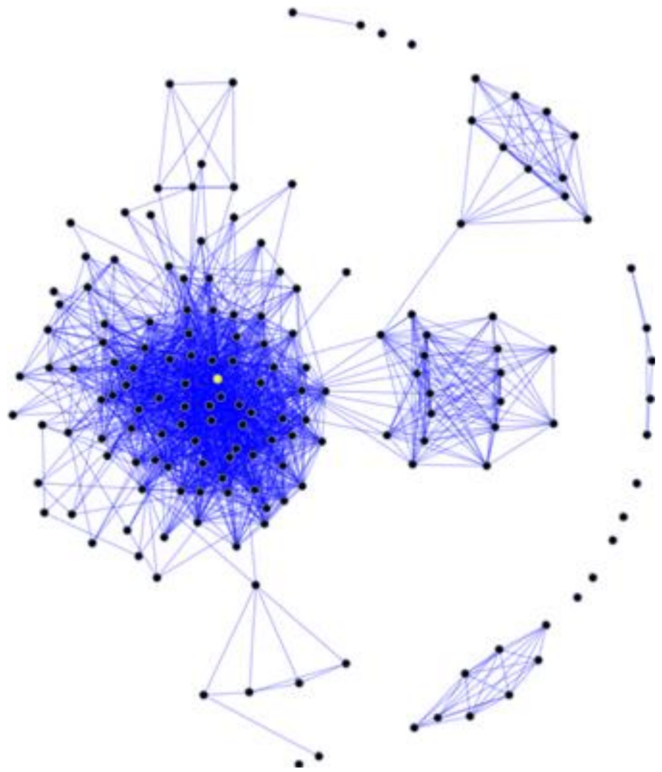


Vaswani et al. '17

Relationships in Data

So far, all of our data consists of points

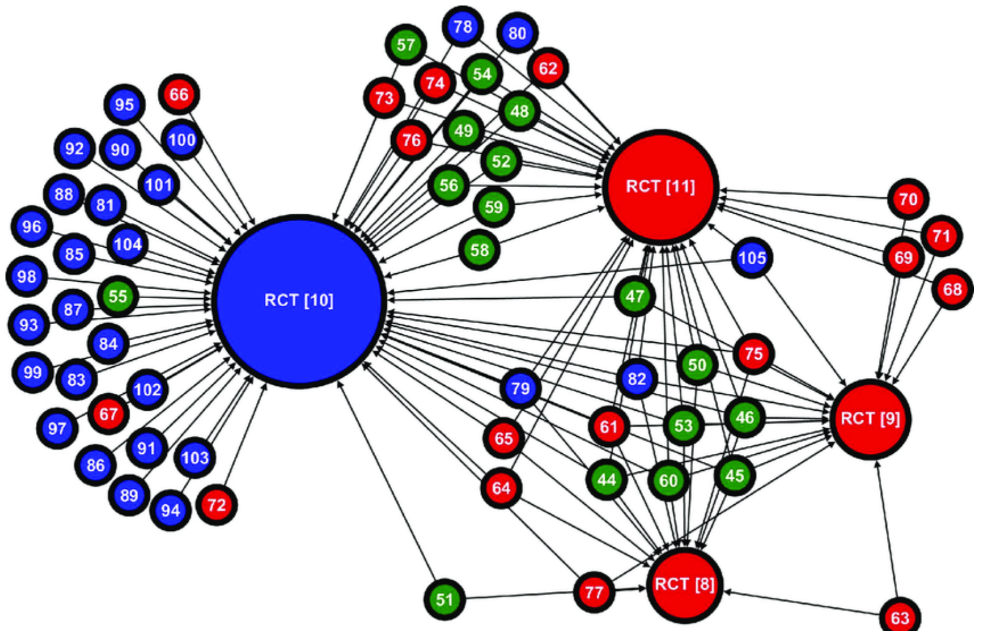
- Assume all are independent, "unrelated" in a sense $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$
- Pretty common to have relationships between points
  - **Social networks**: individuals related by friendship
  - **Biology/chemistry**: bonds between compounds, molecules
  - **Citation networks**: Scientific papers cite each other


Wiki

Graph Neural Networks: Motivations

- We'll do this via **"graph neural networks"**
- Canonical dataset: **citation networks**.
  - Instances are scientific papers
  - Labels: subfield/genre
  - Graphs: if a paper cites another, there's an edge between them

Note: other features as well (text)



Leng

Graph Neural Networks: Approach

- **Idea**: want to use the graph information in our predictions.
- **Semi-supervised aspect**: don’t need all the graph’s nodes to be labeled---use network to predict unlabeled nodes.
  - We’ll see **much more** of this for foundation models!
- Traditional approach: GNNs keep hidden state at a node

$$\underset{\substack{\uparrow \\ \text{Node embedding at time step t}}}{\mathbf{h}_v^{(t)}} = \sum_{u \in N(v)} f(\underset{\substack{\uparrow \\ \text{Node features}}}{\mathbf{x}_v}, \underset{\substack{\uparrow \\ \text{Edge features}}}{\mathbf{x^e}_{(v,u)}}, \mathbf{x}_u, \underset{\substack{\uparrow \\ \text{Neighbor embedding at previous step}}}{\mathbf{h}_u^{(t-1)}})$$

Scarselli et al: “The graph neural network model”, 2009.

Graph Convolutional Networks (GCNs)

Have: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n), G = (V, E)$

How should our new architecture look?

- Still want layers
  - linear transformation + non-linearity
- Now want to integrate neighbors
- Bottom: graph convolutional network

Parameters

Non-Linearity

Hidden Layer Representation

$$H^{(\ell+1)} = \sigma(H^{(\ell)} W^{(\ell)})$$

$$\Big\downarrow$$

$$H^{(\ell+1)} = \sigma(A_G H^{(\ell)} W^{(\ell)})$$

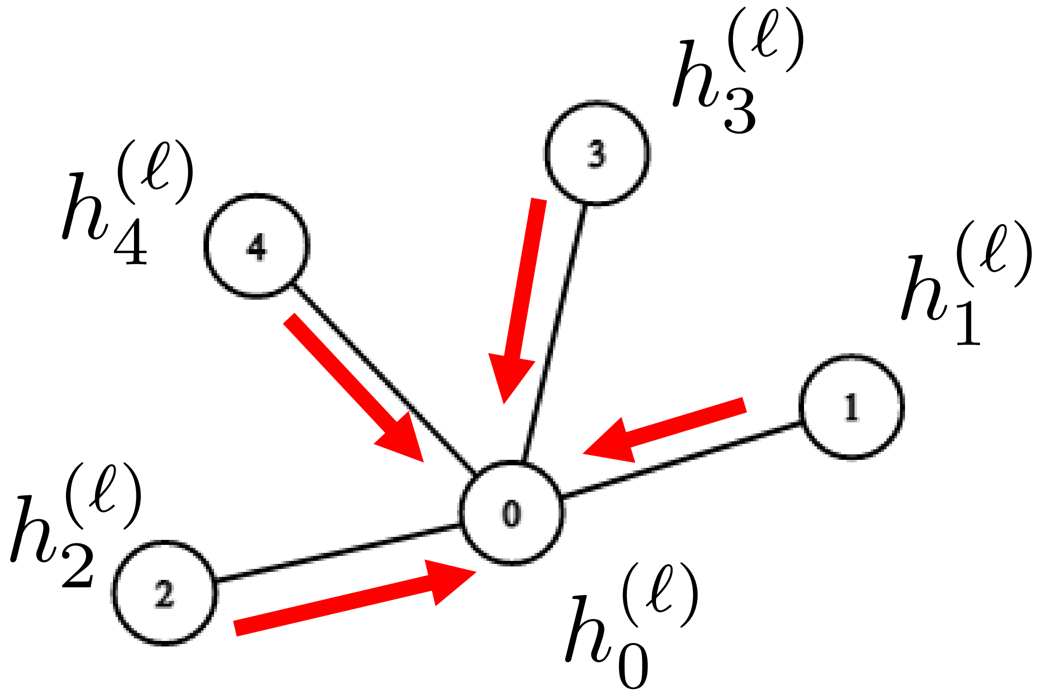
Graph Mixing

Kipf and Welling: "Semi-Supervised Classification with Graph Convolutional Networks"

Graph Convolutional Networks (GCN)

Let's examine the GCN architecture in more detail

- Difference: "graph mixing" component
- At each layer, get representation at each node
- Combine node's representation with neighboring nodes
- "**Aggregate**" and "**Update**" rules



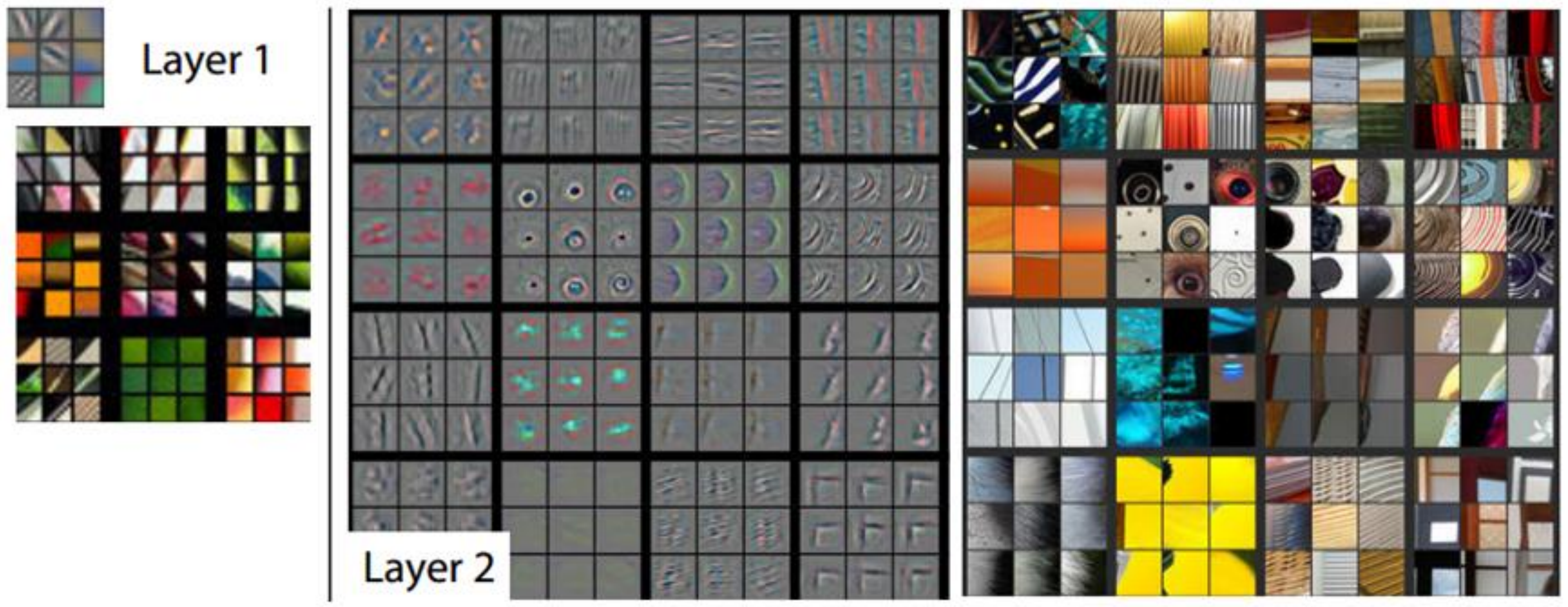
Outline

- **Sequence Models & Graph Models**
  - Recurrent neural networks, architecture, LSTMs, graphs and graph-based models
- **Self-Supervised Learning**
  - Motivation, basic idea, masking, autoencoders, student-teacher methods
- **Contrastive Learning**
  - Intuition, losses, getting positives and negatives, frameworks

Break & Questions

Representations

- Basic idea in ML is to discover useful representations
  - I.e., higher level features that are discriminative
  - These are not necessarily present in raw data…



Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labled Layer 2, we have representations of the 16 different filters (on the left)

Desphande

Where to Get Representations?

- Deep learning:
  - Automatically obtain good features, but
  - **Downside**: Need lots of labeled data

- Pre-trained models:
  - E.g., ResNets trained on ImageNet. Use last layer (pre-prediction)
  - **Downside**: pre-trained task may not match our goal task

- Generative model encoders:
  - **Downside**: may not relate to semantics we care about

Representations from **Self Supervision**

- There’s lots of information in our dataset already
  - Of course, specific to our task

- Need to create tasks from unlabeled data: “Pretext tasks”
  - Ex: predict stuff you already know

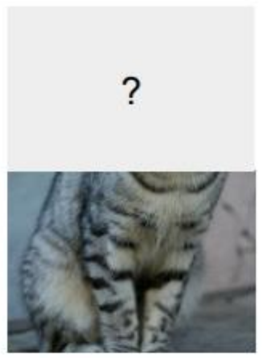
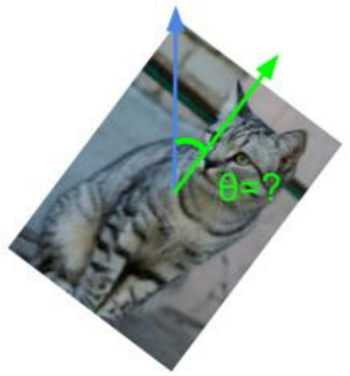
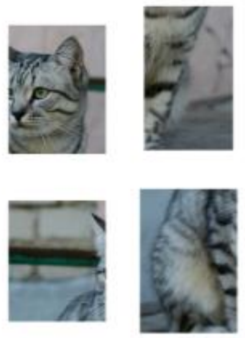


image completion



rotation prediction

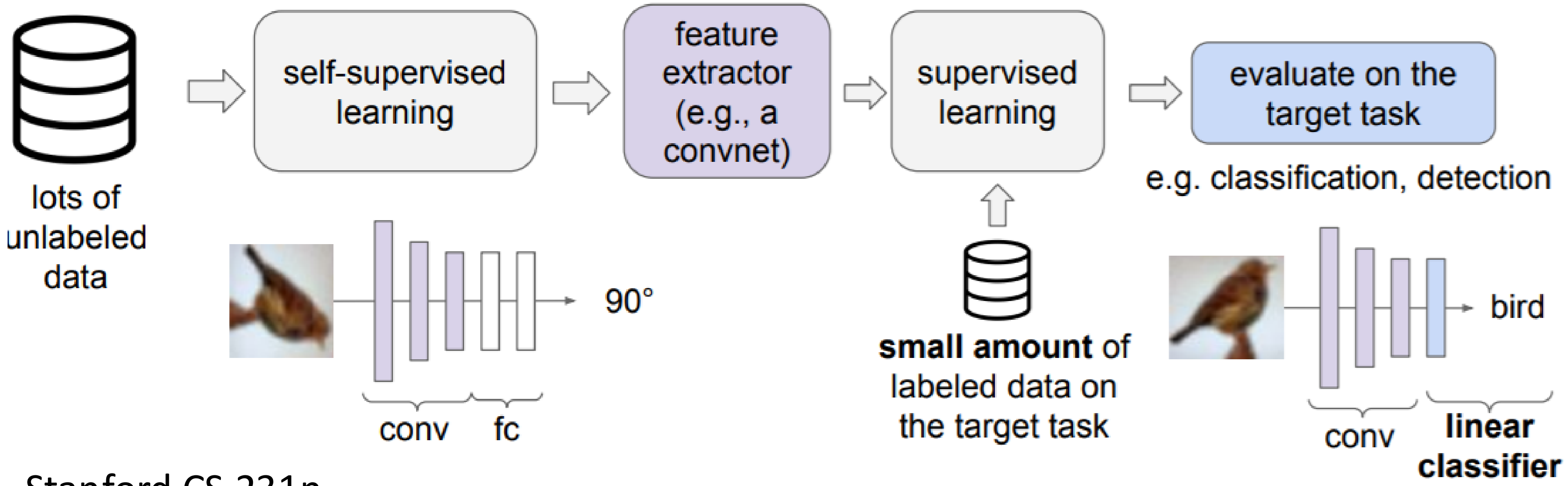

“jigsaw puzzle”

colorization

Stanford CS 231n

Using the Representations

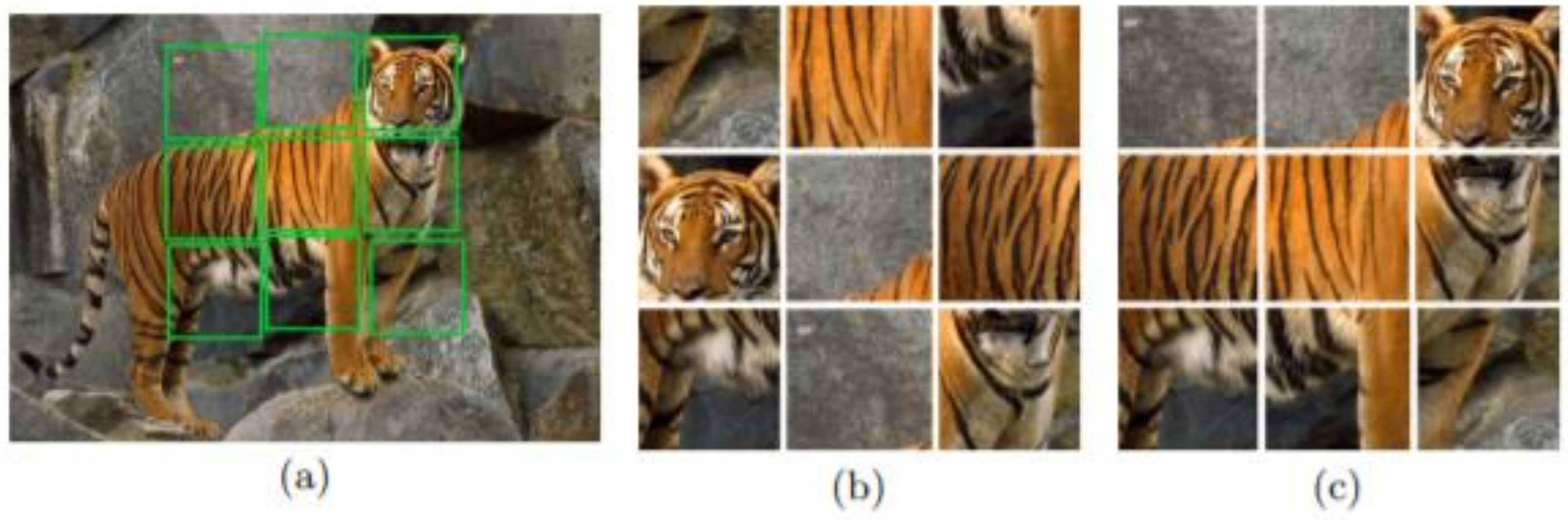
- Don’t care specifically about our performance on self-task
- Use the learned network as a feature extractor
- Once we have labels for a particular task, train
  - A small amount of data


lots of unlabeled data
self-supervised learning
feature extractor (e.g., a convnet)
supervised learning
evaluate on the target task
e.g. classification, detection
90°
conv
fc
small amount of labeled data on the target task
bird
conv
linear classifier

Stanford CS 231n

Self Supervision: Pretext Tasks

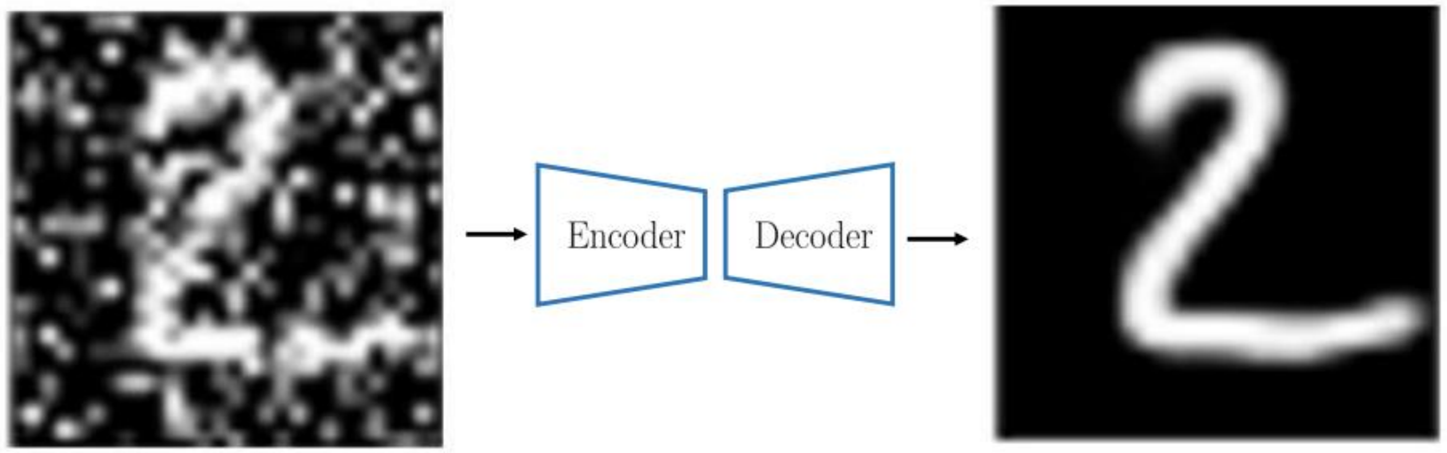
- Lots of options for pretext tasks
  - Predict rotations
  - Coloring
  - Fill in missing portions of the image
  - Solve puzzles:



Noroozi and Favaro

Pretext Tasks

- Choice of **task** matters…
  - Classic approach, image denoising, may not provide ideal results:

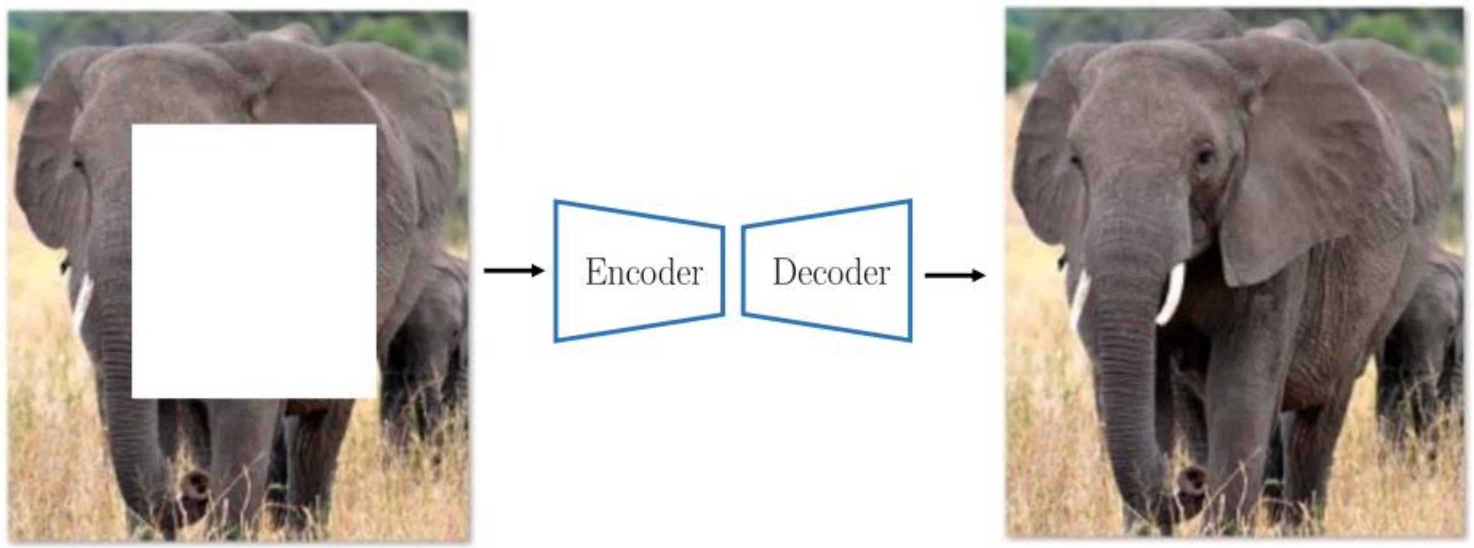


Vincent et al 2008

  - Why? Might be too easy, representations don’t need to pick up anything “deep”

Pretext Tasks

- Choice of **task** matters…
  - What about predicting masked boxes?

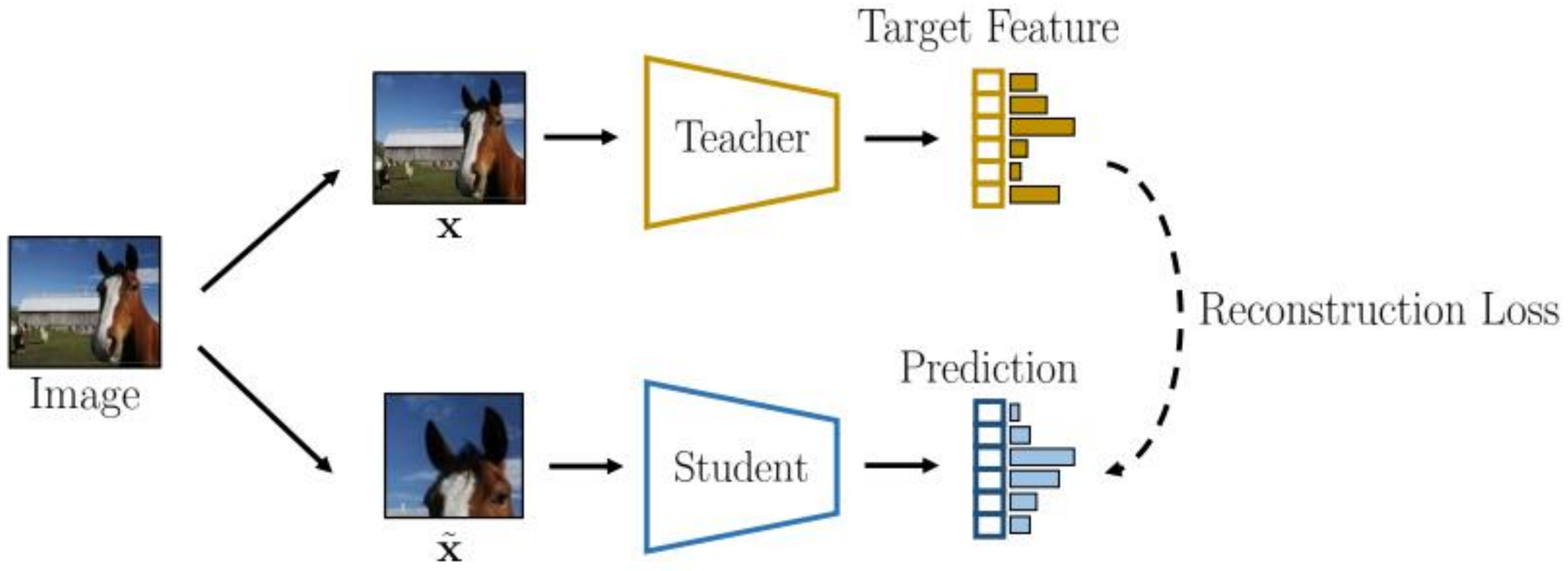


Pathak et al 2016

  - Better, but might focus on picking up **irrelevant** details

Teacher-Student Methods

- Another way to proceed: teacher/student architecture
  - Teacher produces a representation
  - Student tries to match it from a different 'view'



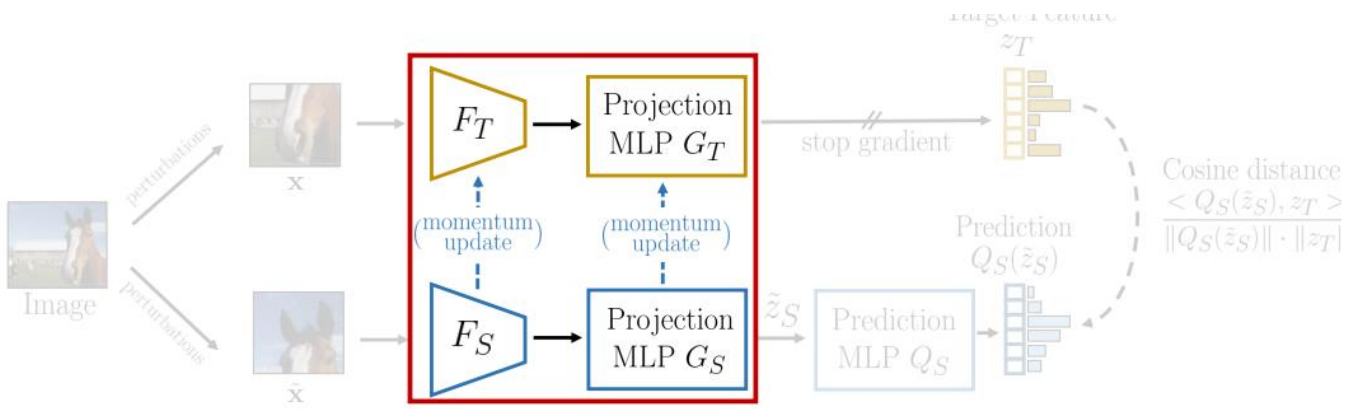
Gidaris & Bursuc, 2021.

Teacher-Student Method Variations

- Another way to proceed: teacher/student architecture
  - Teacher produces a representation
  - Student tries to match it from a different 'view'

Many variations

- If student networks is smaller: **distillation** approach
- Special update rules for interactions between teacher/student models (BYOL)



Use exponential moving average for online updating the teacher at each training step:

$$\theta_{\mathrm{T}}^{(t)} \leftarrow \alpha \cdot \theta_{\mathrm{T}}^{(t-1)} + (1-\alpha) \cdot \theta_{\mathrm{S}}^{(t)}$$

θ_{T} : teacher parameters

θ_{S} : student parameters

Outline

- **Sequence Models & Graph Models**
  - Recurrent neural networks, architecture, LSTMs, graphs and graph-based models
- **Self-Supervised Learning**
  - Motivation, basic idea, masking, autoencoders, student-teacher methods
- **Contrastive Learning**
  - Intuition, losses, getting positives and negatives, frameworks

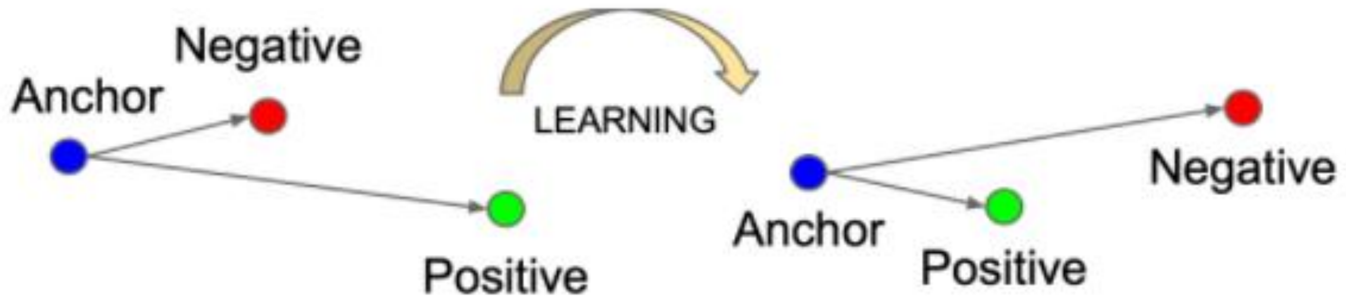
Using Multiple Samples

- So far, each sample considered individually
  - Didn’t use relationships between samples
  - We saw a lot of power in doing this for graph data/tasks

- Leads to **contrastive learning**
  - **Idea:** reps. for similar samples should be close, for unrelated samples, should be distant
  - We don’t have access to labels, but samples with same class (‘positives’) should have close reps., others (‘negatives’) far

Training with Positive and Negative Samples

- Leads to **contrastive learning**
  - **Idea:** We don’t have access to labels, but samples with same class (‘positives’) should have close reps., others (‘negatives’) far

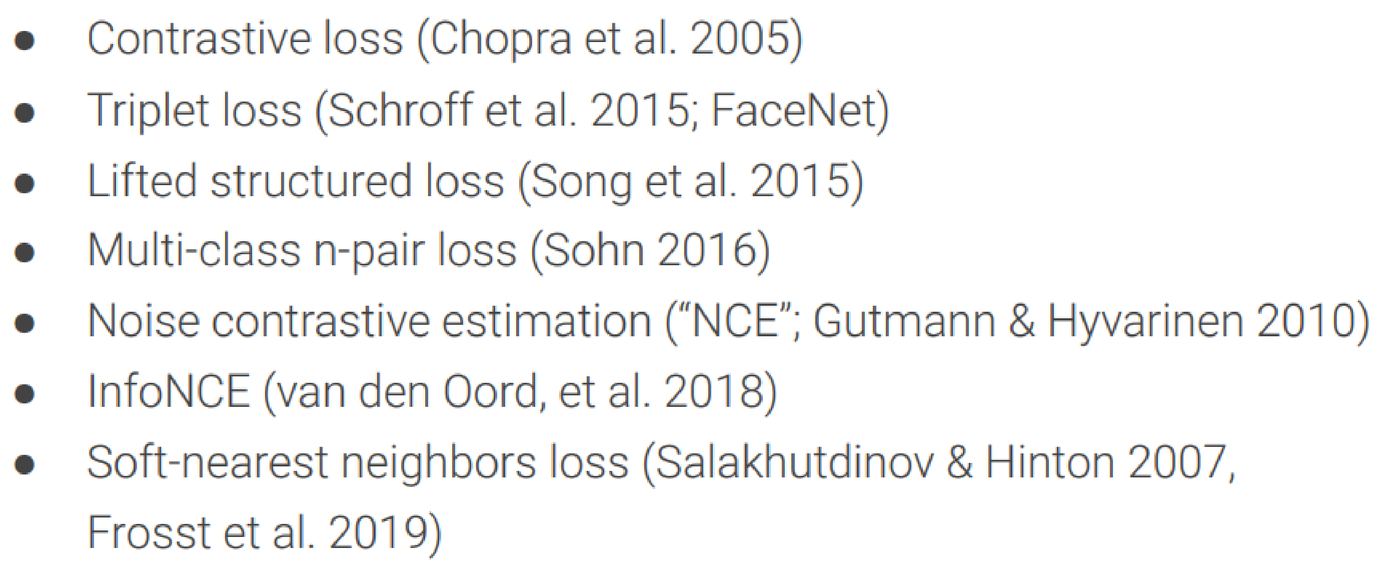


Schroff et al. 2015

- **Two questions:**
  - What loss shall we pick?
  - How do we get positive and negative samples (**no labels**!!) ?

Contrastive Losses

- What **loss** shall we pick?
- A whole zoo of losses



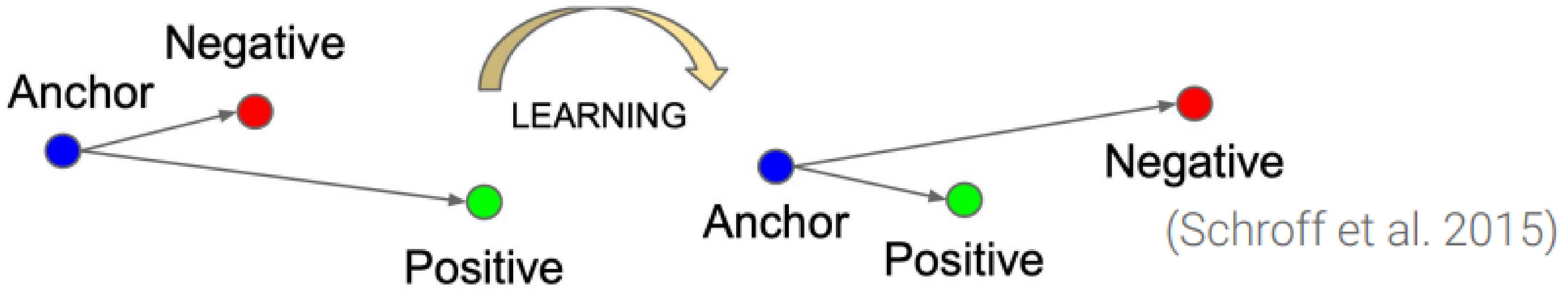
- Contrastive loss (Chopra et al. 2005)
- Triplet loss (Schroff et al. 2015; FaceNet)
- Lifted structured loss (Song et al. 2015)
- Multi-class n-pair loss (Sohn 2016)
- Noise contrastive estimation ("NCE"; Gutmann & Hyvarinen 2010)
- InfoNCE (van den Oord, et al. 2018)
- Soft-nearest neighbors loss (Salakhutdinov & Hinton 2007, Frosst et al. 2019)

- With a variety of properties… let's see a couple of examples.

Triplet Loss

- Squared distance formulation
  - Recall our regression loss---same basic principle

$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x}\in\mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$



InfoNCE Loss

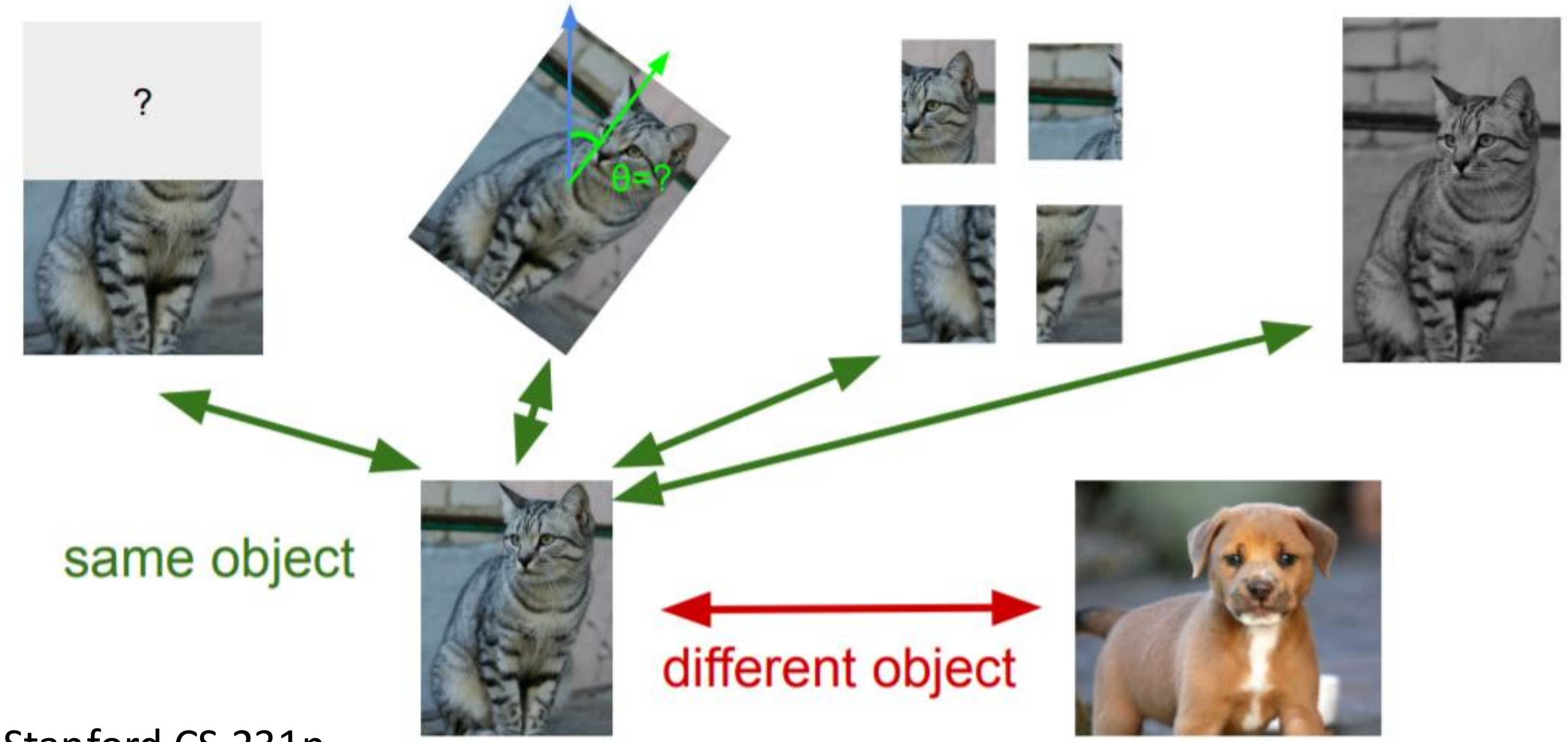
- Info Noise Contrastive Estimation Loss
  - Formulation based on logistic regression
  - Note: operates with 1 **positive** sample, k **negative** samples
  - Builds on earlier NCE, permits multiple negatives

$$L = -E_X\left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{k-1} \exp(s(f(x), f(x_j^-))}\right]$$

Van den Oord et al., 2018

Generating **Positives**

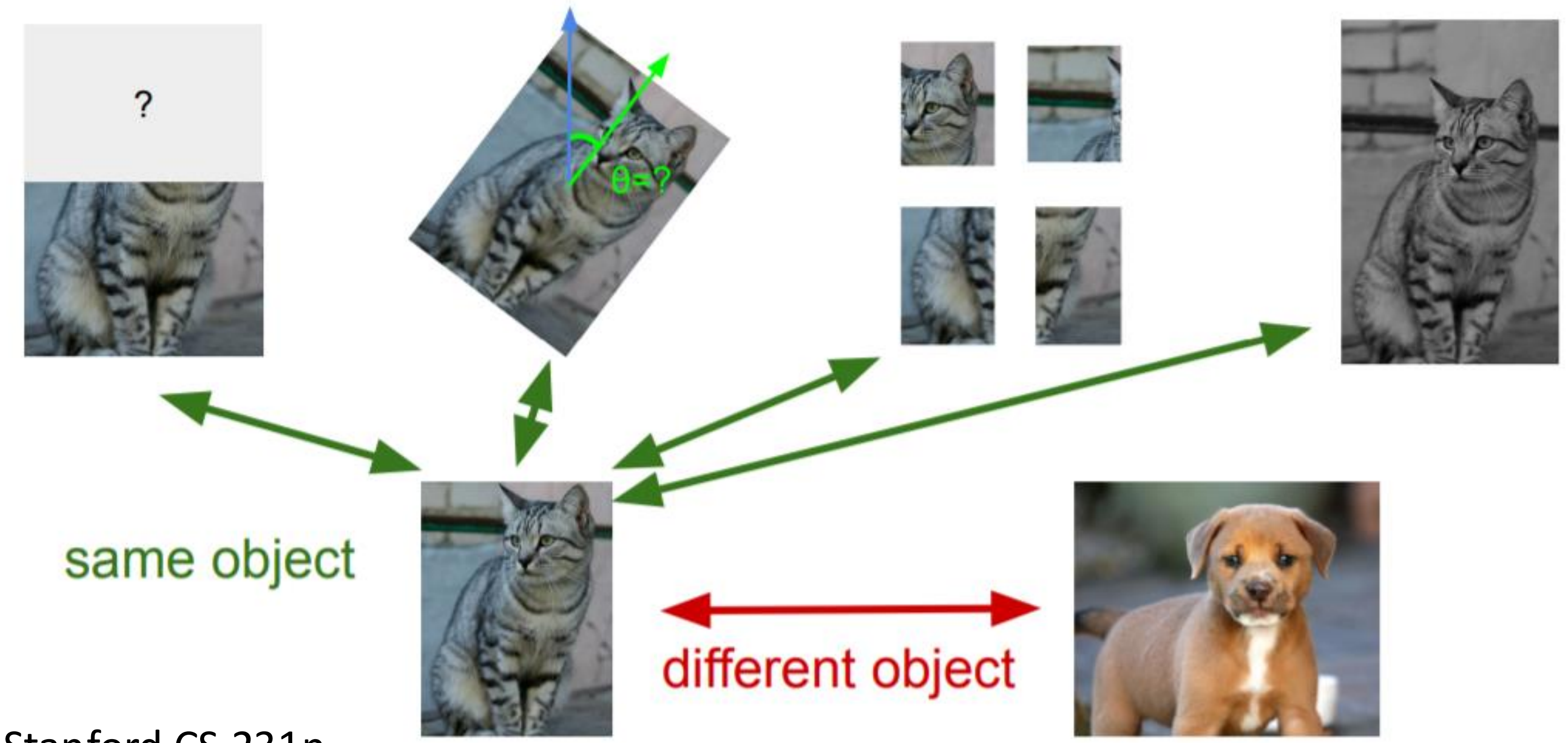
- Easiest way to generative **positives**
  - Augmentations/transformations (recall last lecture)
  - Borrow from existing literature on data augmentation



Stanford CS 231n

Generating **Negatives**

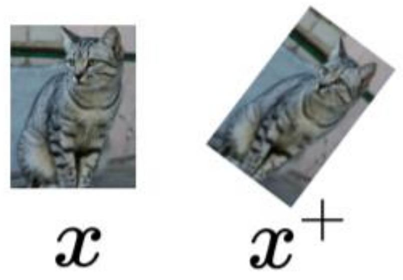
- Generally harder
  - Trivial way: sample at random. Will be wrong sometimes
  - Harder: try to mine negatives; use pre-trained models to make more likely. Active research area



Putting it Together: Contrastive Training

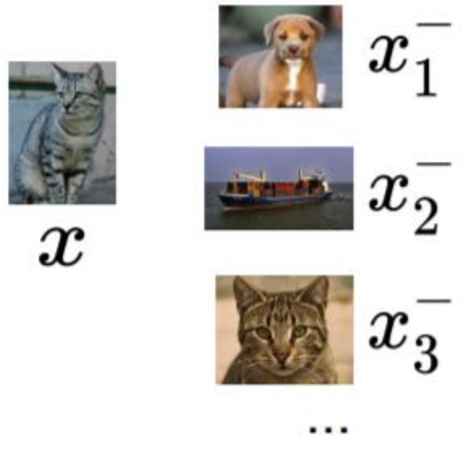
- Pick a sample, generate transformed versions, sample multiple negatives, train the loss:
  - Say, InfoNCE

$$L = -E_X \left[\log \frac{\exp(s(f(x), f(x^+))}{\exp(s(f(x), f(x^+)) + \sum_{j=1}^{k-1} \exp(s(f(x), f(x_j^-))} \right]$$



Positive sample: keep close

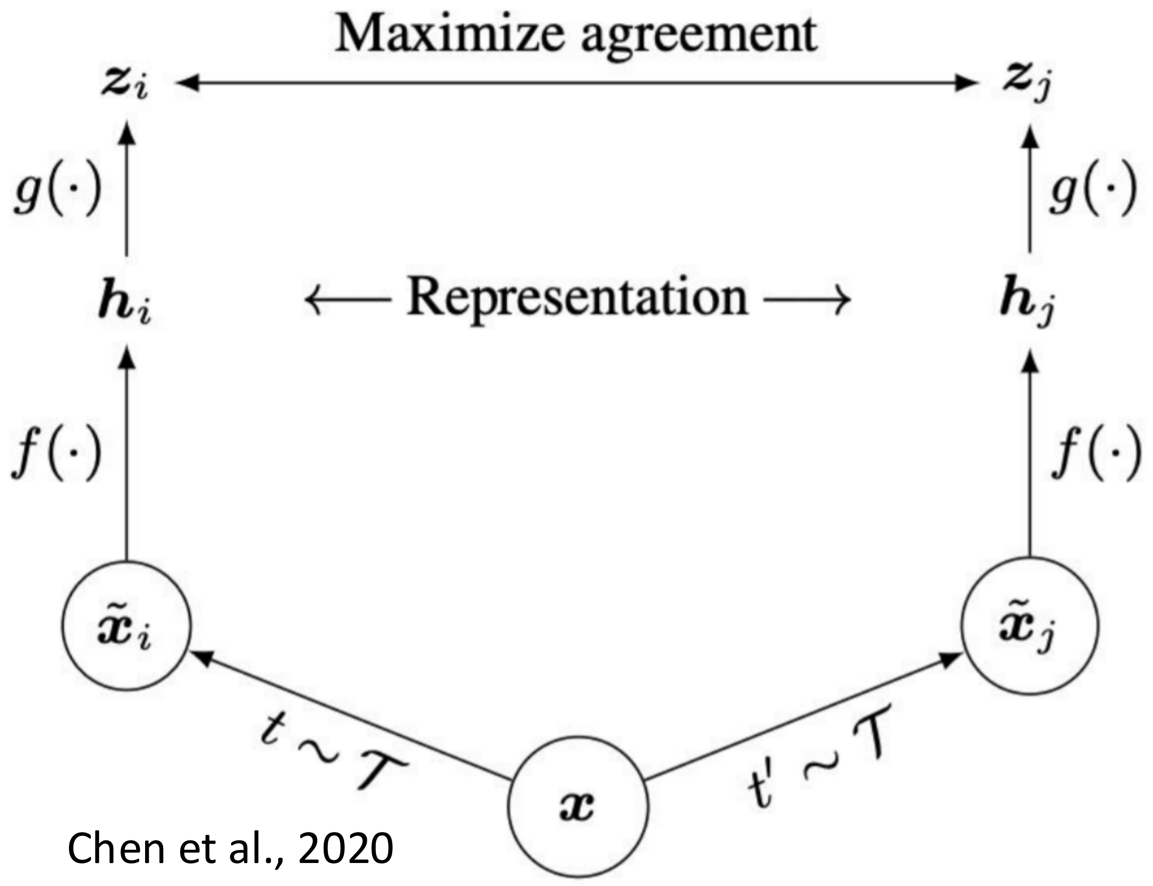
Negative sample keep far



Contrastive Learning: Frameworks

- Many approaches (very active area of research)
  - A popular approach: SimCLR. Score function is cosine similarity,

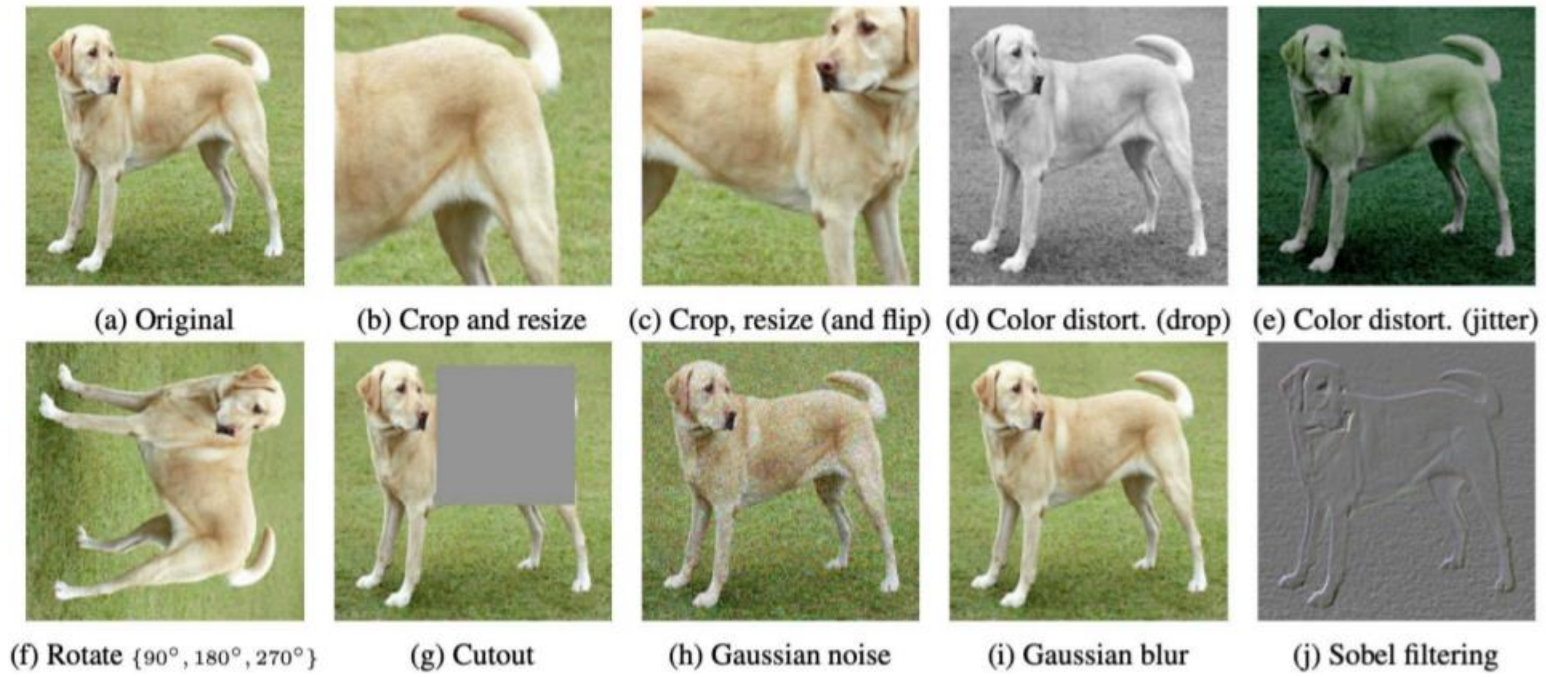
  - Generate positive samples:
  Choose random augmentations



Chen et al., 2020

Contrastive Learning: Frameworks

- Many approaches (very active area of research)
  - A popular approach: SimCLR. Score function is cosine similarity,
  - Generate positive samples:
    Choose random augmentations


(a) Original
(b) Crop and resize
(c) Crop, resize (and flip)
(d) Color distort. (drop)
(e) Color distort. (jitter)
(f) Rotate {90°, 180°, 270°}
(g) Cutout
(h) Gaussian noise
(i) Gaussian blur
(j) Sobel filtering