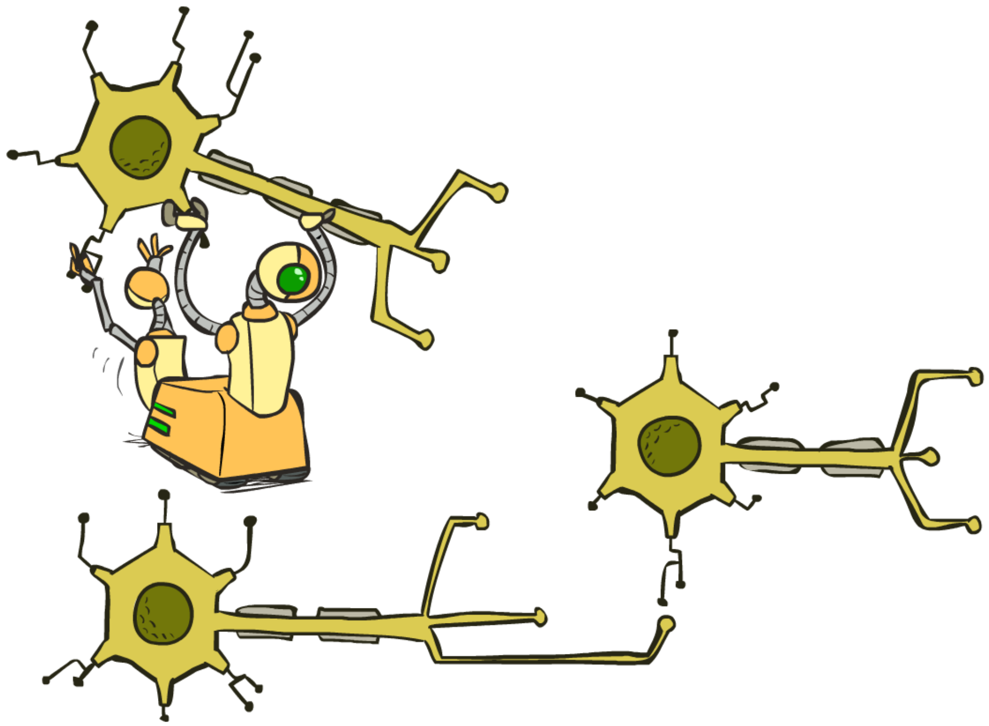
CS 188: Artificial Intelligence

Special Topics: NLP/CV/RL



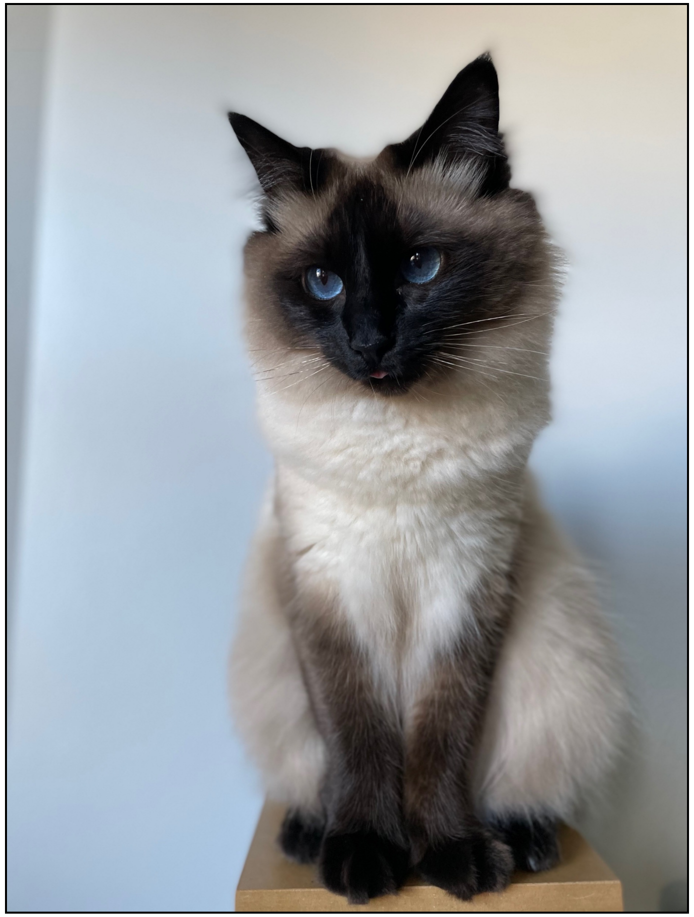
Instructor: Nicholas Tomlin

[Slides courtesy of Dan Klein, Abigail See, Greg Durrett, Yejin Choi, John DeNero, Eric Wallace, Kevin Lin, Fei-Fei Li, Sergey Levine, Pieter Abbeel, and many others]

What tasks do we care about?

- Object detection and classification
- Semantic segmentation
- Image captioning
- Visual question answering
- Video classification and understanding
- Image generation
- …

Image Classification



cat
dog
horse
person
airplane
house
…

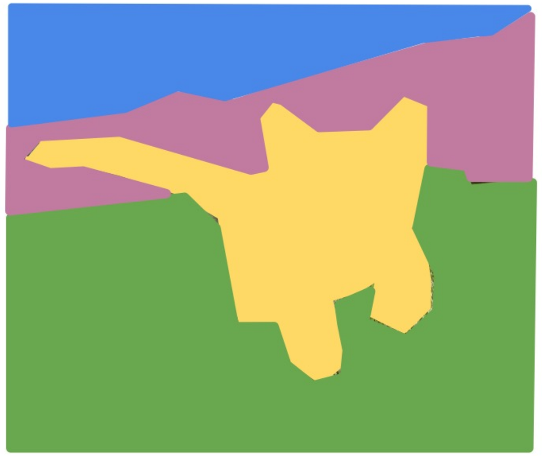
Beyond Image Classification

Classification

CAT

No spatial extent

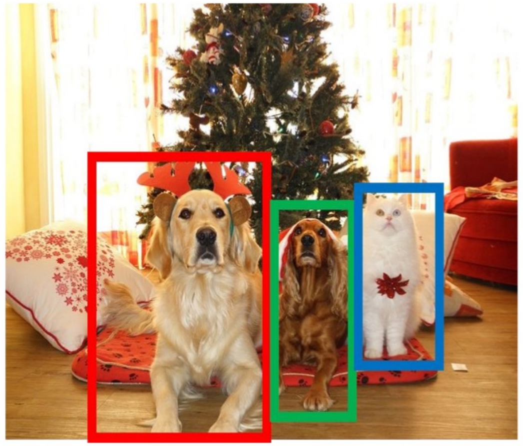
Semantic Segmentation



GRASS, CAT, TREE, SKY

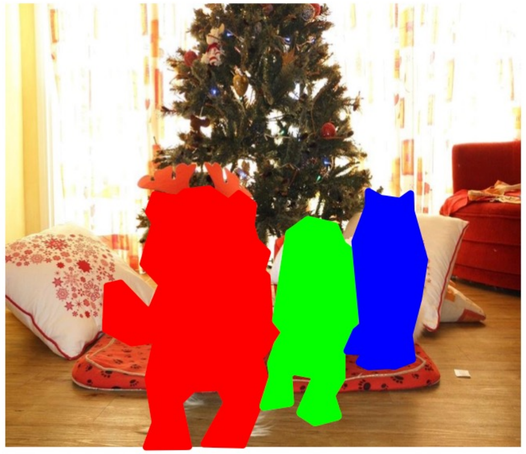
No objects, just pixels

Object Detection



DOG, DOG, CAT

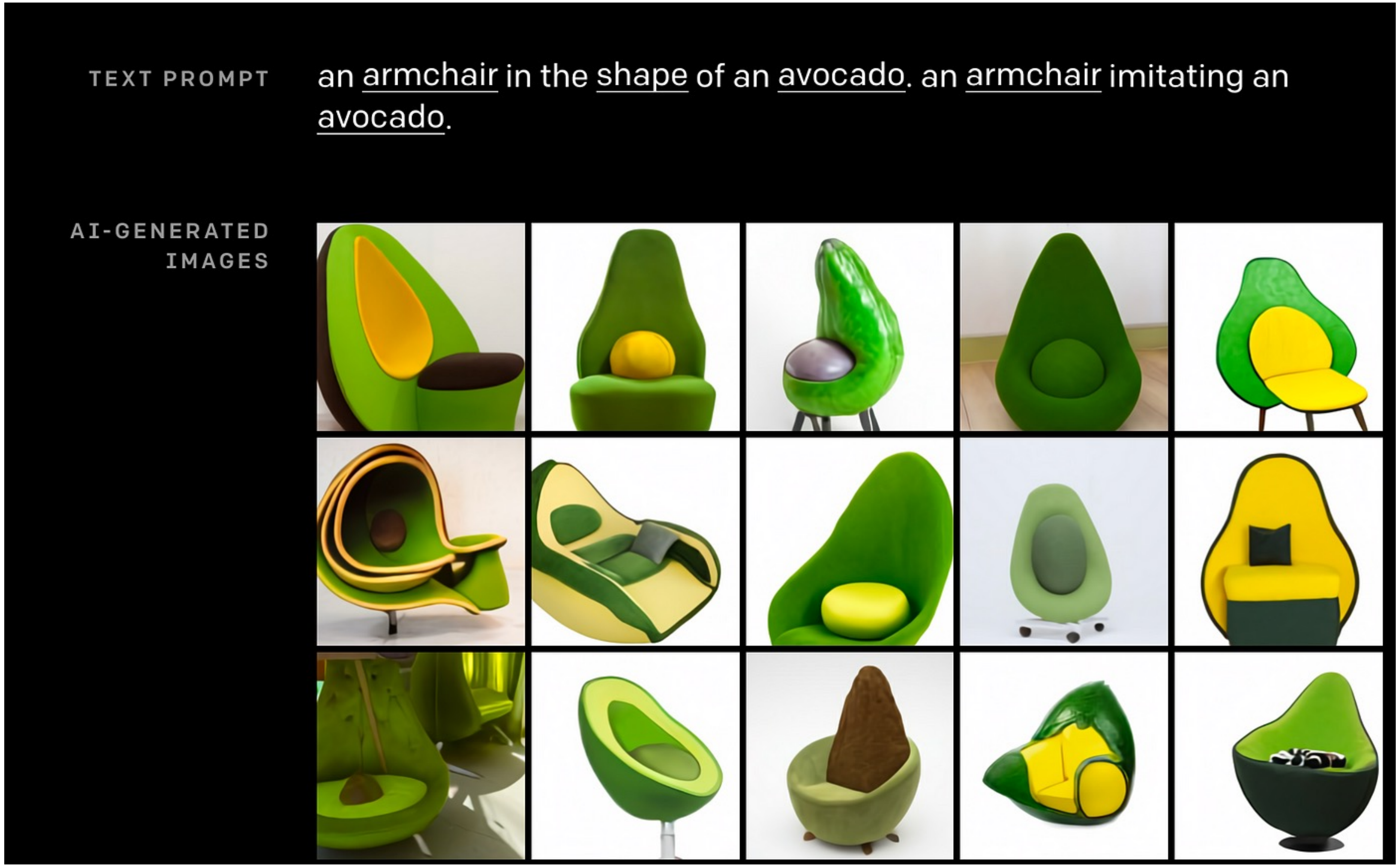
Instance Segmentation



DOG, DOG, CAT

Multiple Object

Image Generation



Recall: MNIST Digit Classification

Task specification:

- Input features: binary pixel values
- Output: a digit classification (0-9)

Issues with Naïve Bayes classifier:

- Can overfit to individual pixels
- Not robust to scaling, movement left/right, etc.

0

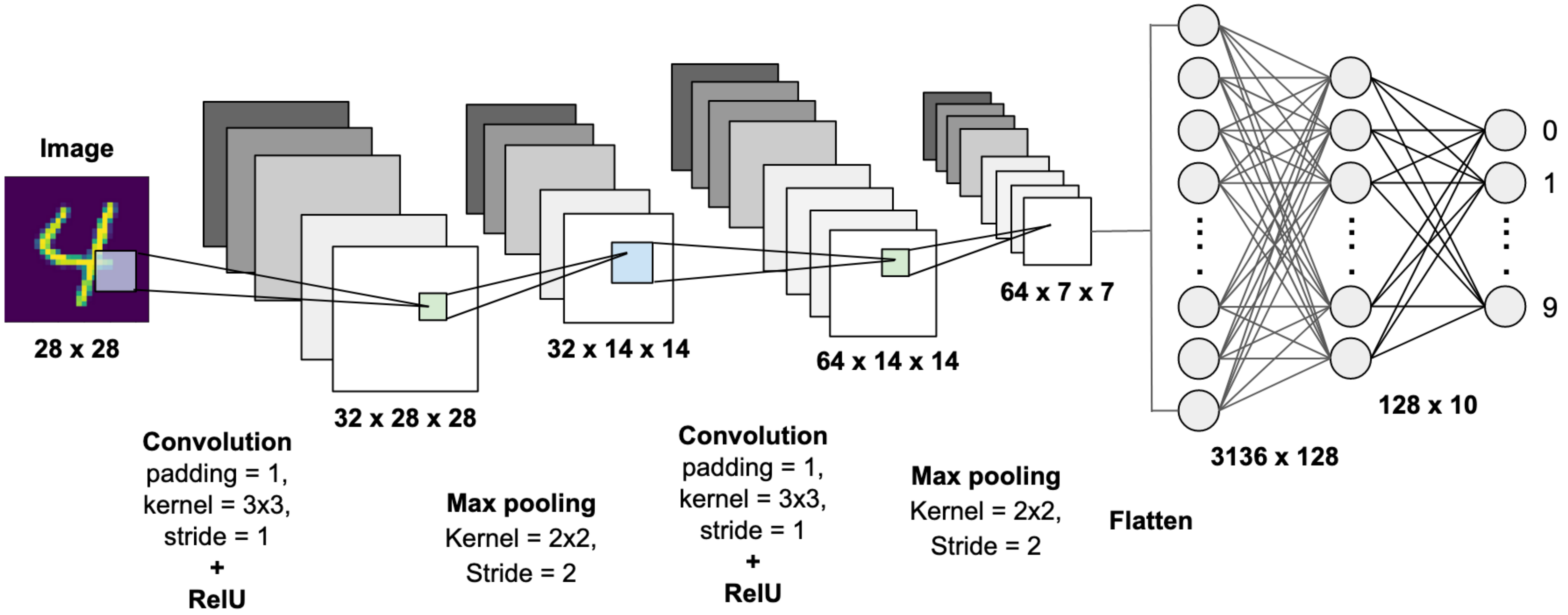
1

2

1

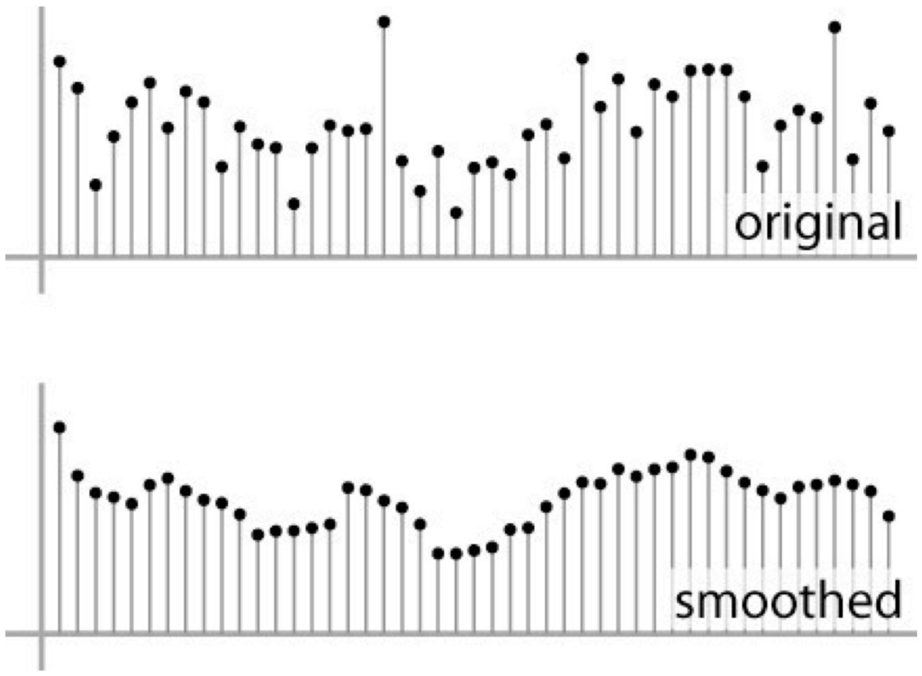
??

Convolutional Neural Networks



Convolution in 1D

- Basic idea: define a new function by averaging over a sliding window
- Example in one dimension: smoothing



Convolution in 1D

- Moving average:

$$c[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} a[j]$$

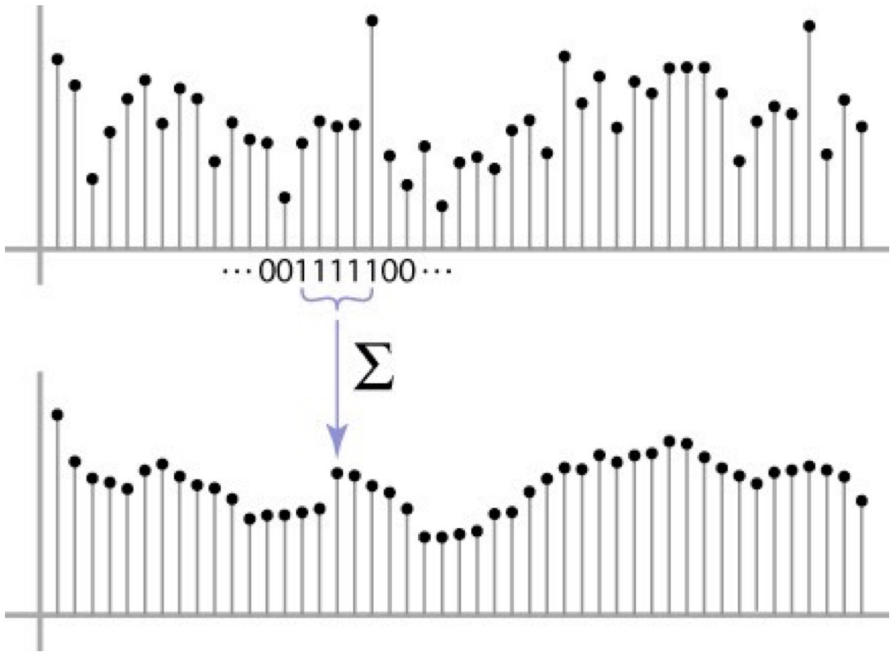
- Convolution: same idea but with weighted average

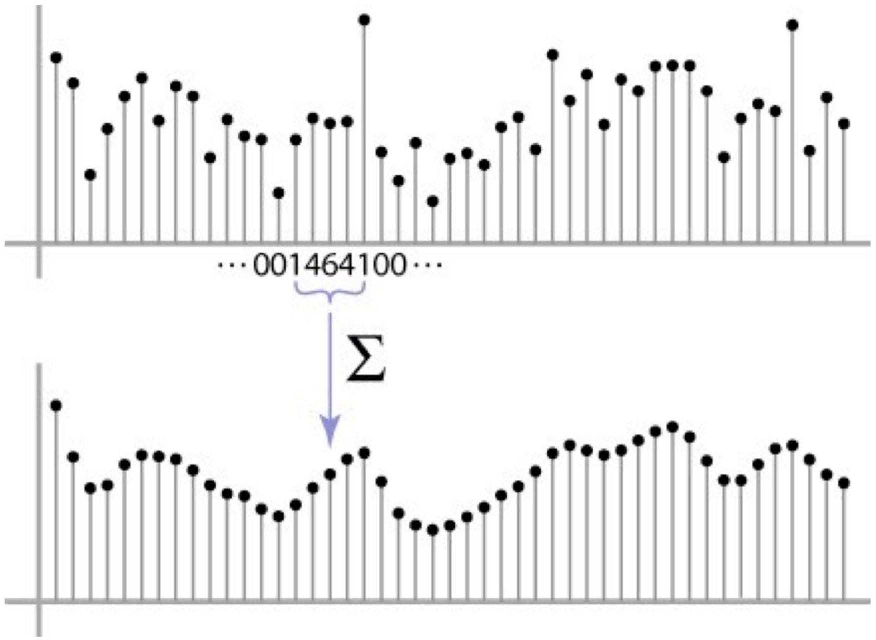
$$(a \star \boxed{b})[i] = \sum_{j} a[j] \cdot b[i-j]$$

called a *filter*

Convolution in 1D

- Filters in one dimension:
  - Box filter: [..., 0, 0, 1, 1, 1, 1, 1, 0, 0,...]/5
  - Gaussian filter: [..., 0, 0, 1, 4, 6, 4, 1, 0, 0,...]/16



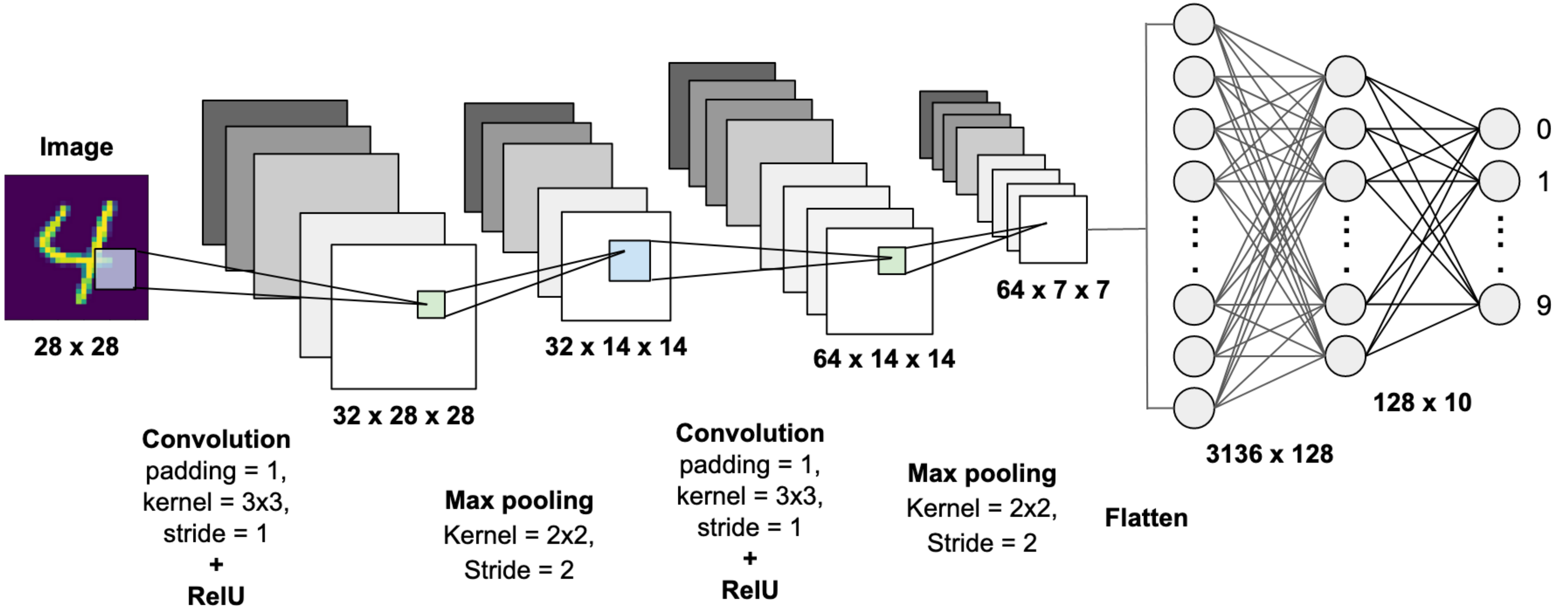


Convolution in 2D

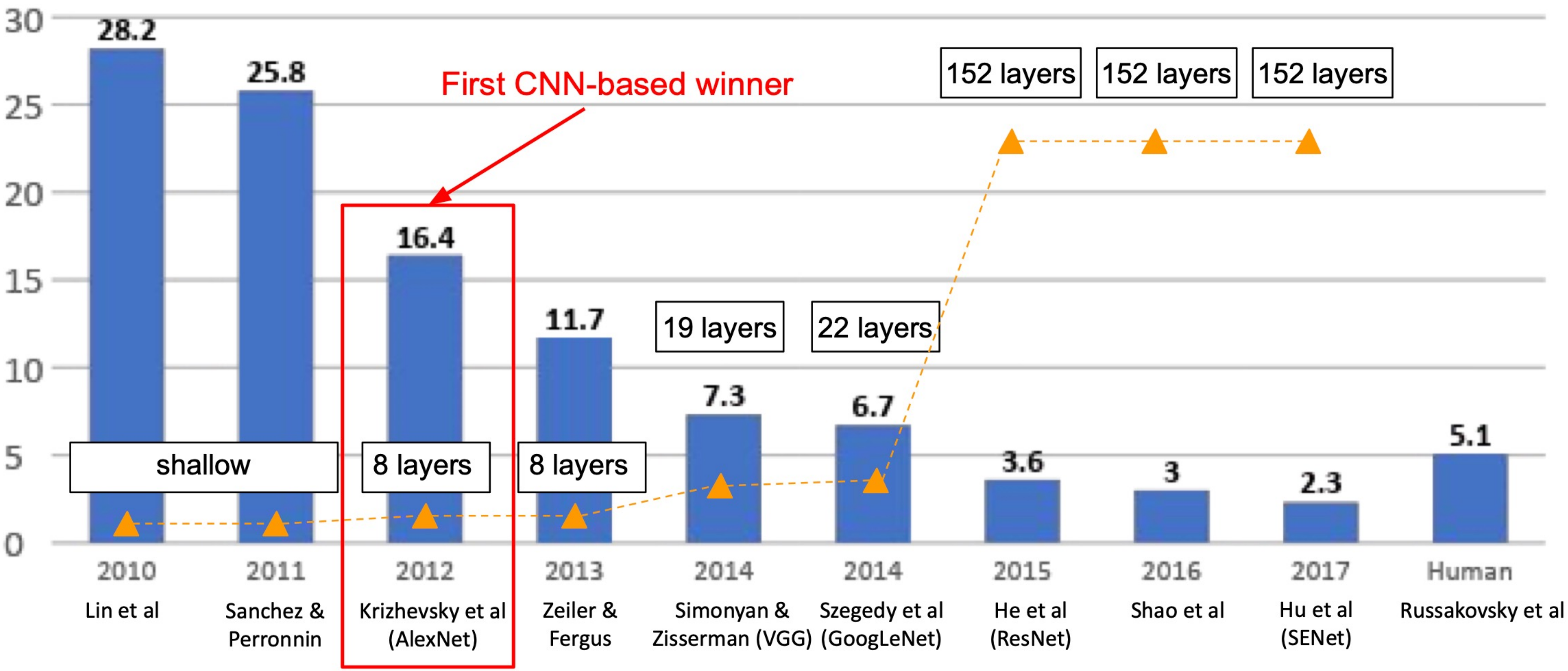
- Filters in two dimensions: same idea but apply over a square patch of inputs (often 3x3 or 5x5)
- Applications:
  - Blurring
  - Sharpening
  - Feature detection
  - …

Convolutional Neural Networks

- Key idea: learn the filter weights via backprop

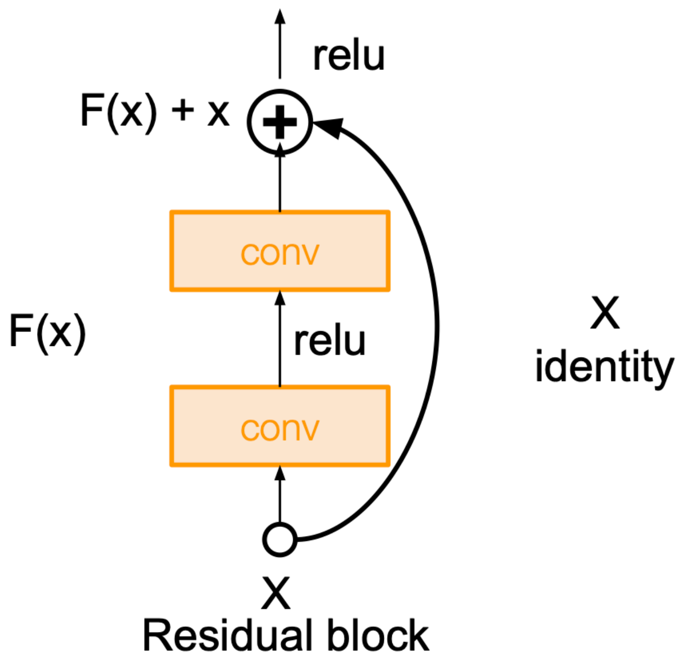


Benchmarking on ImageNet



ResNet (He, et al. 2015)

- Key idea:
  - Want deeper networks with more parameters, but training signal becomes weak
  - Add “skip” connections between layers so that there are shorter paths between early parameters and the final loss function
- ResNet:
  - 152-layer model for ImageNet
  - Massive improvement over all previous CNN-based classification models circa 2015



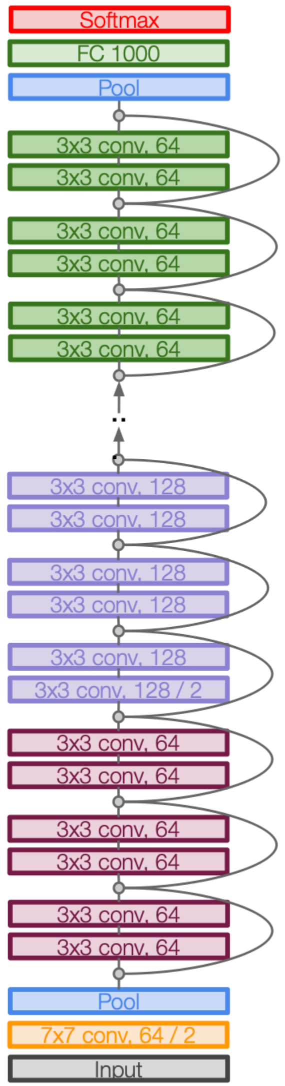
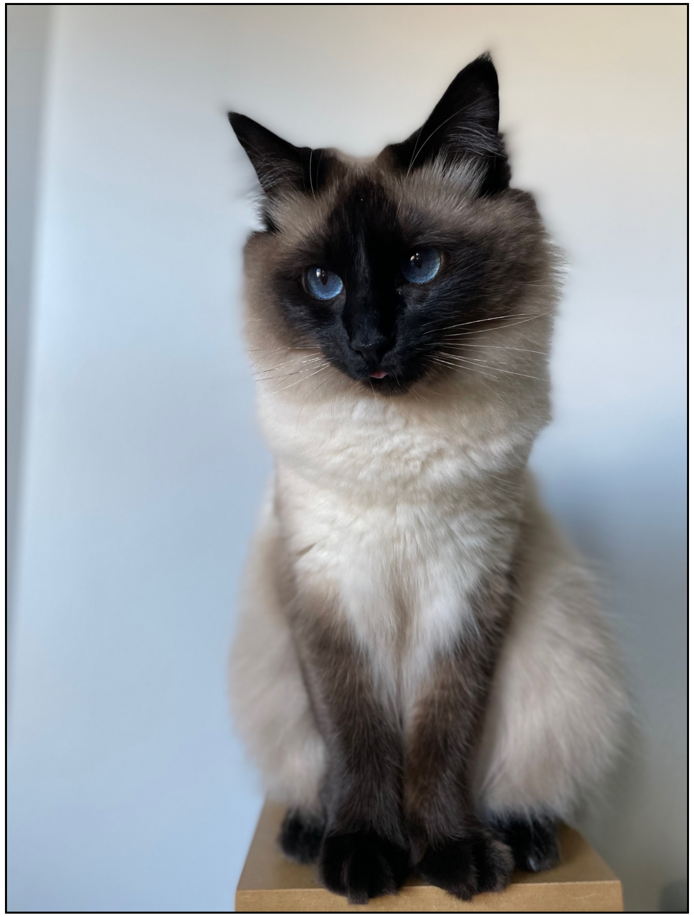
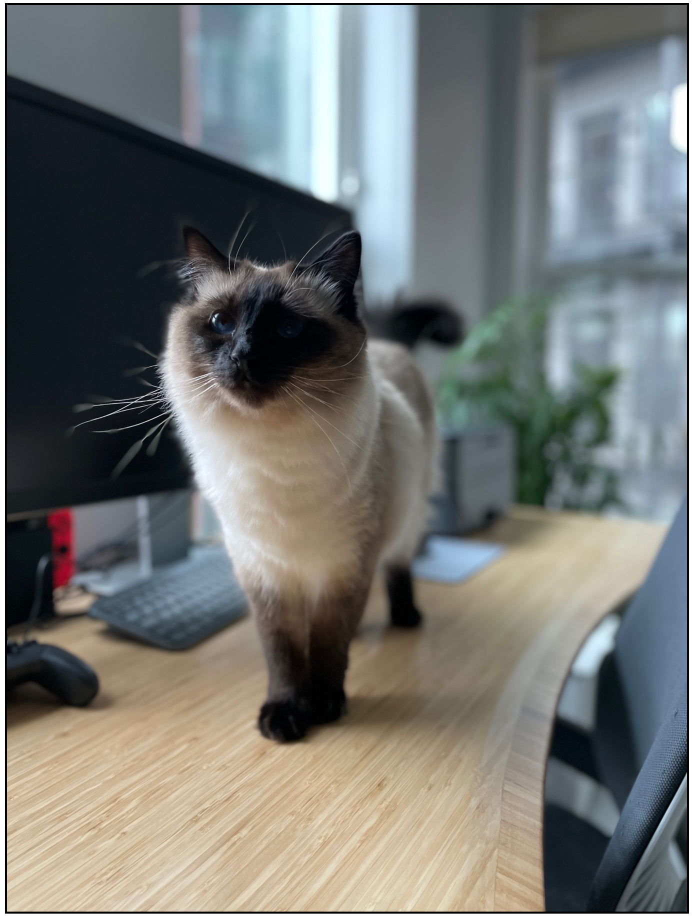


Image Classification



cat
dog
horse
person
airplane
house
…

Image Captioning



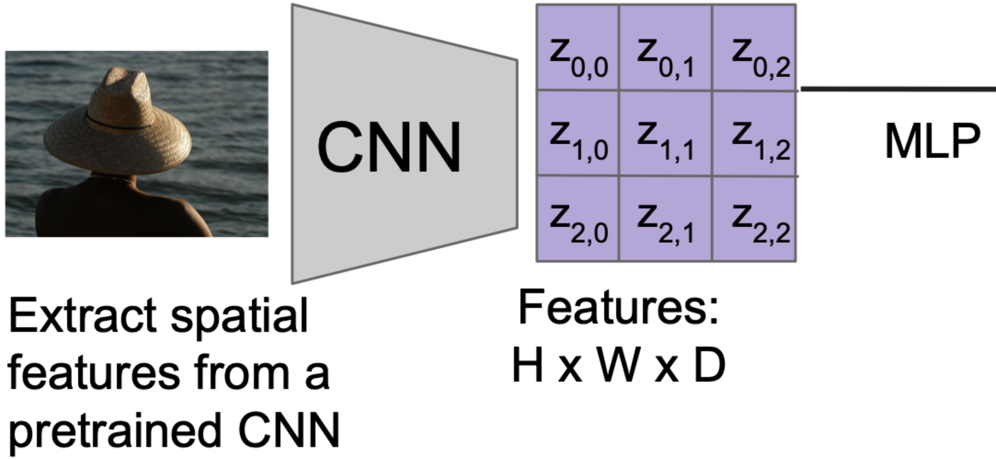
a cat standing on a desk

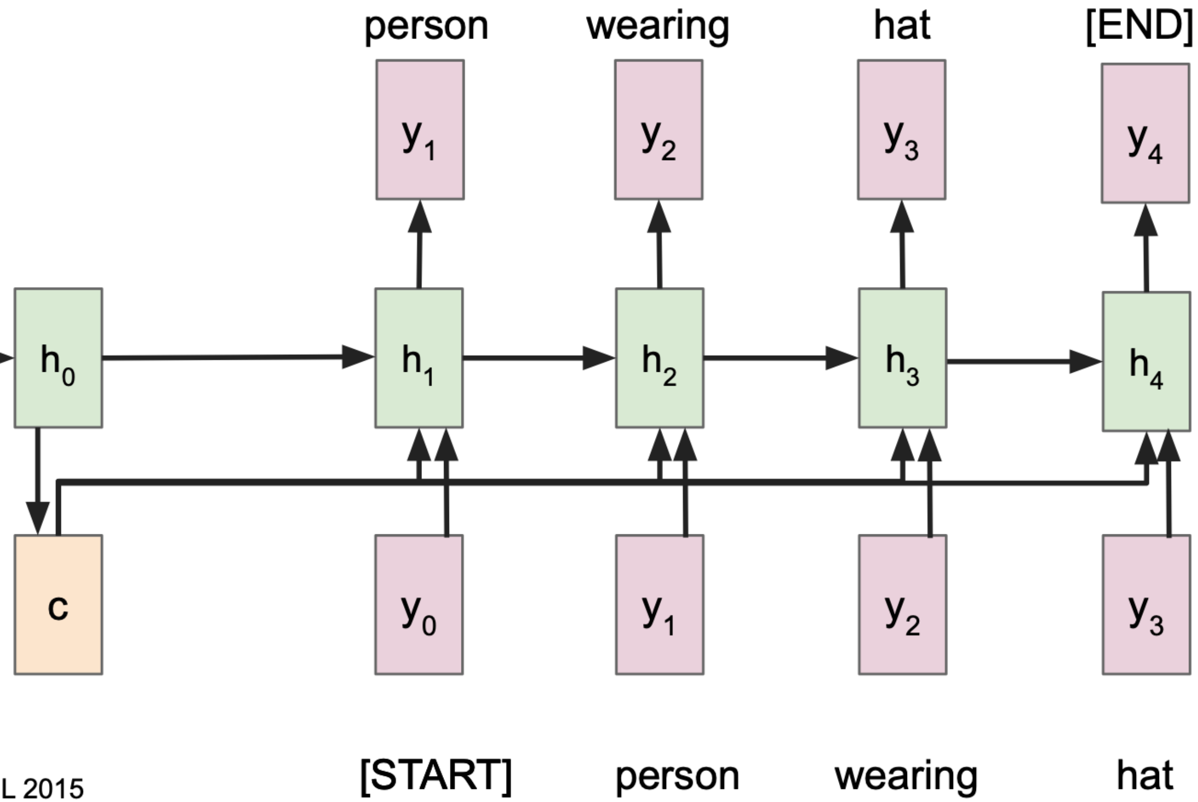
Image Captioning with RNNs

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$

Decoder: $y_t = g_{\mathbf{V}}(y_{t-1}, h_{t-1}, c)$
where context vector c is often $c = h_0$

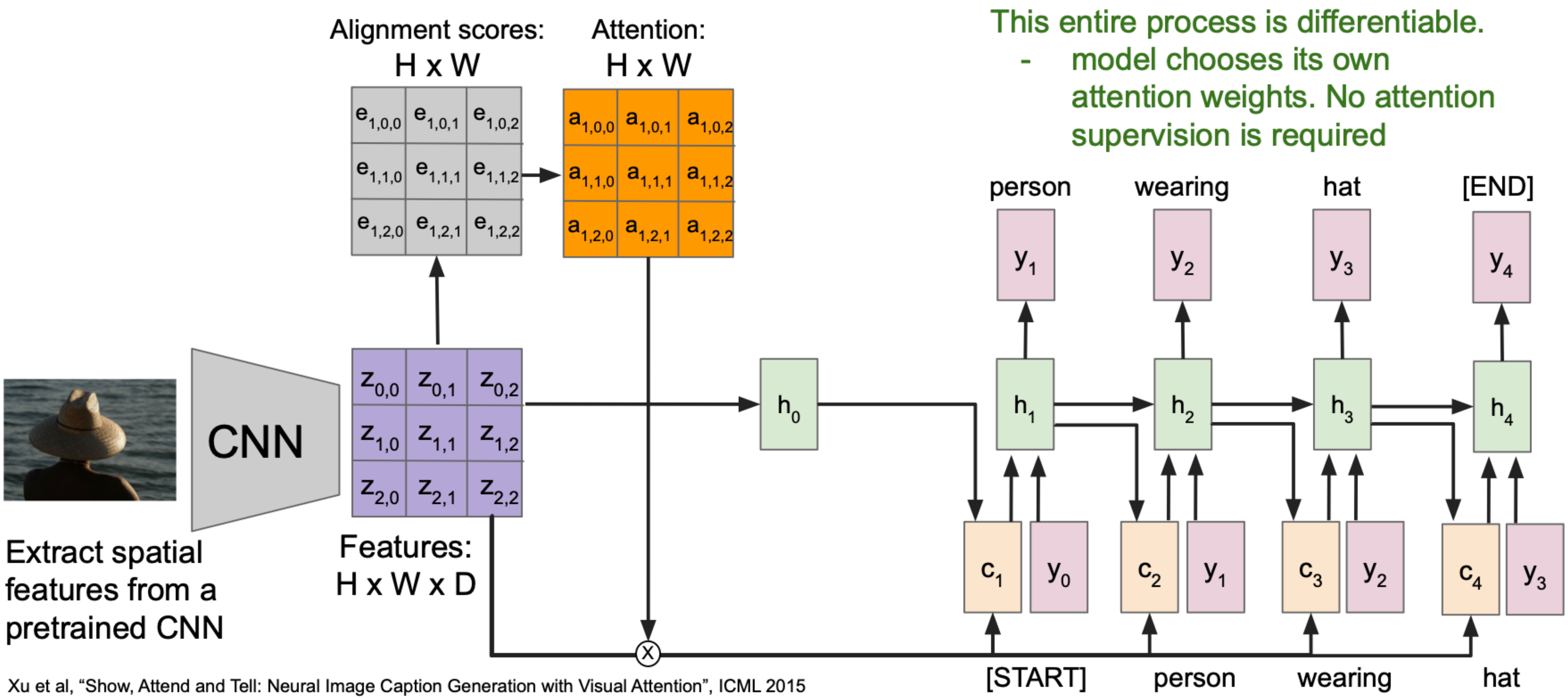
Encoder: $h_0 = f_{\mathbf{W}}(\mathbf{z})$
where **z** is spatial CNN features
$f_{\mathbf{W}}(.)$ is an MLP





Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with RNNs + Attention



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Image Captioning with Transformers

Input: Image **I**
Output: Sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$

Decoder: $y_t = T_D(\mathbf{y}_{0:t-1}, \mathbf{c})$
where $T_D(.)$ is the transformer decoder

Encoder: $\mathbf{c} = T_W(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$T_W(.)$ is the transformer encoder

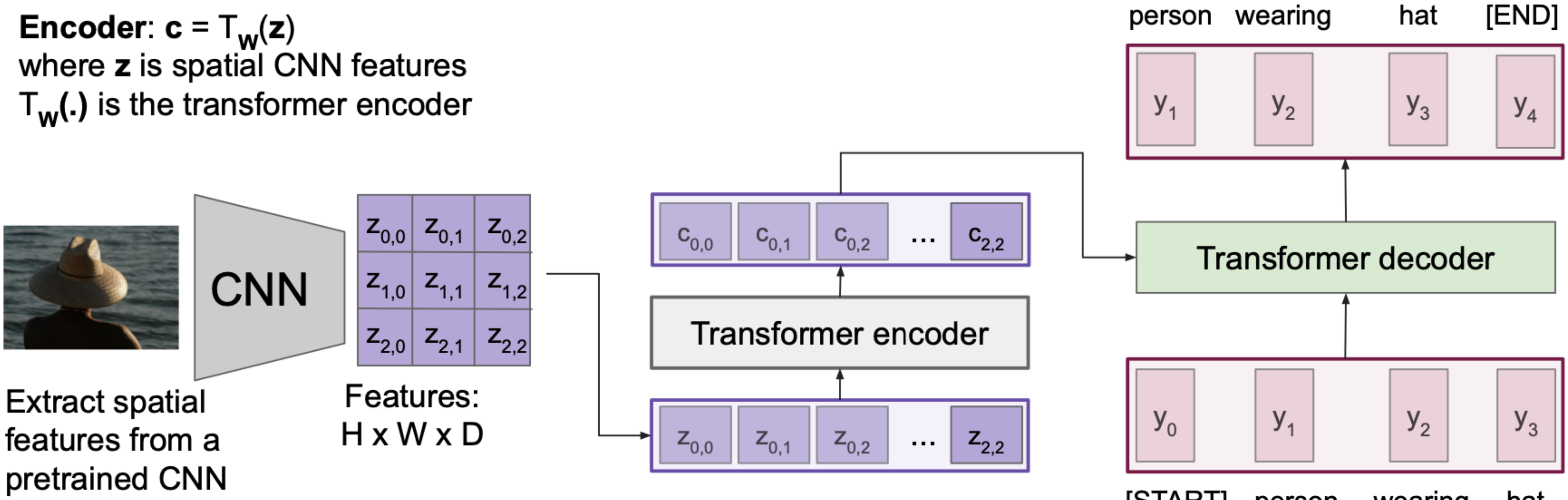
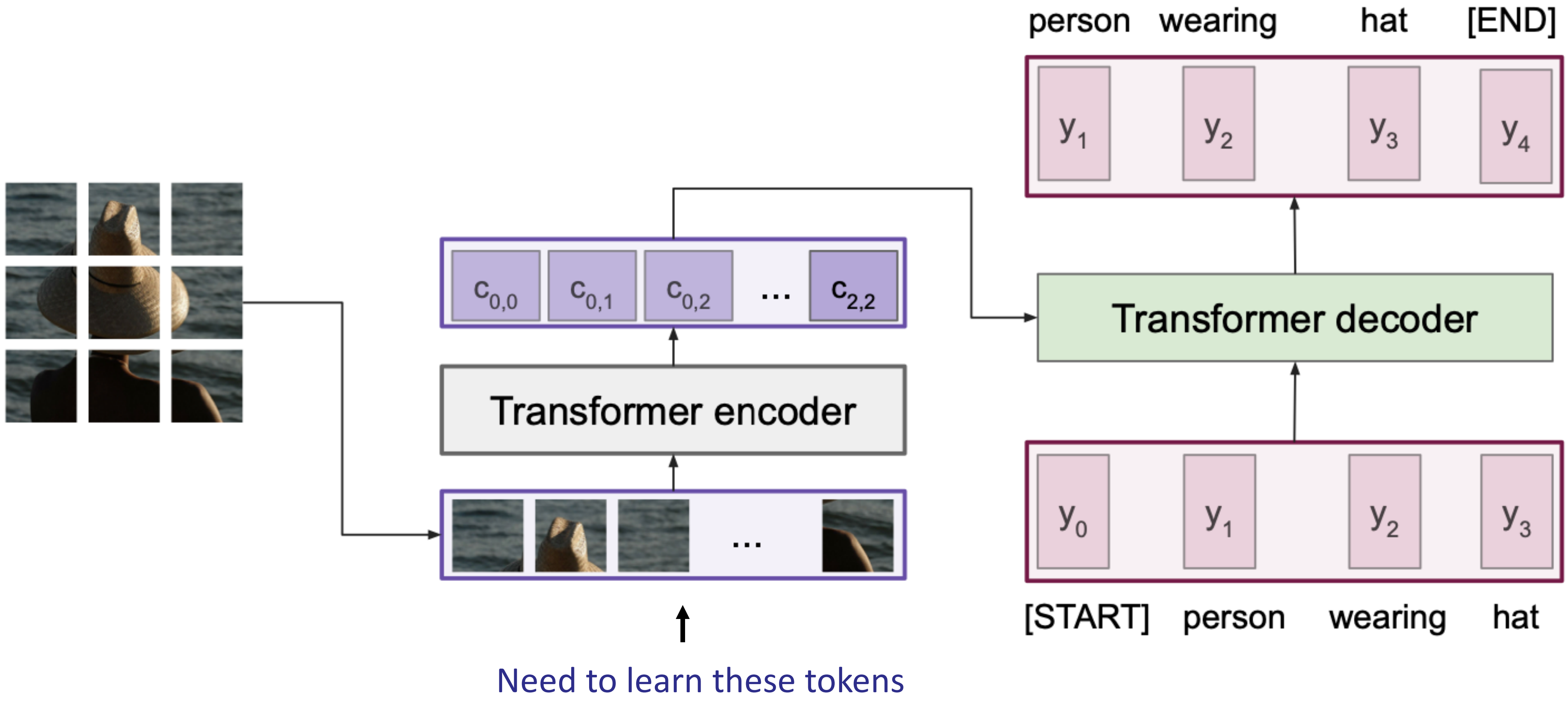
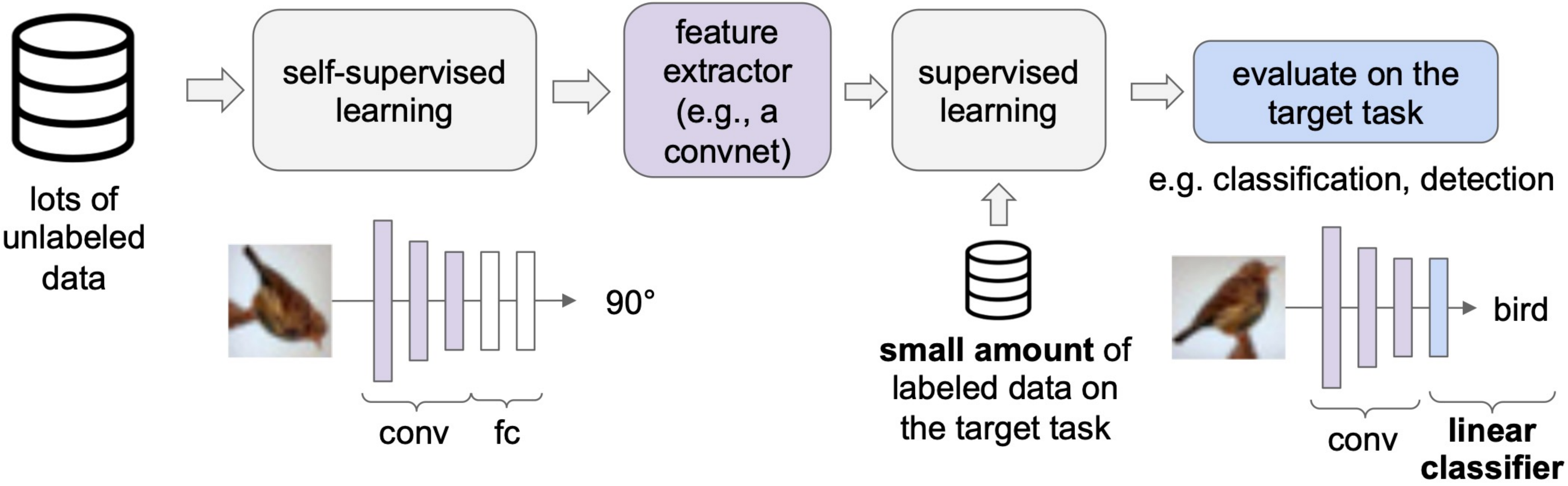


Image Captioning with Vision Transformers



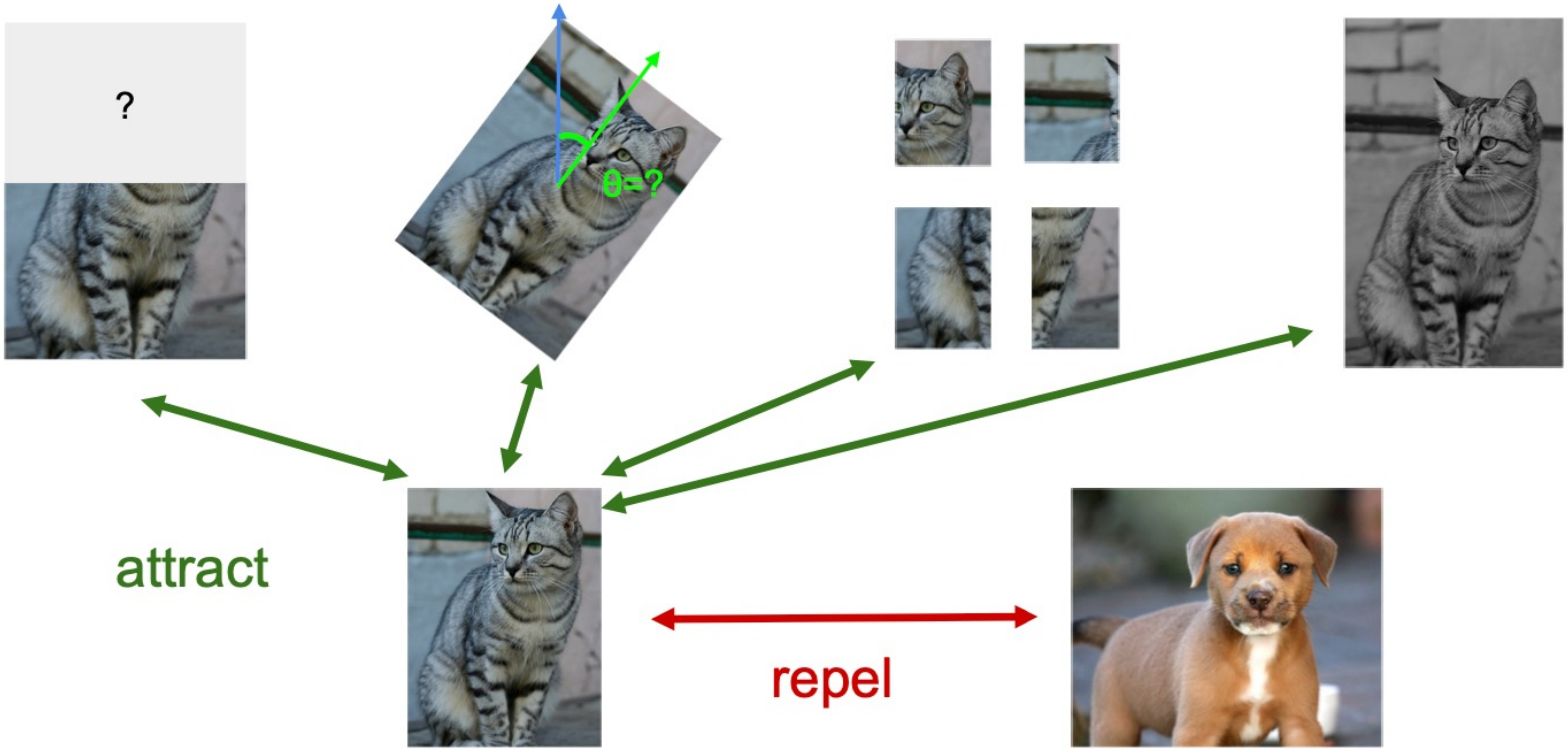
Representation Learning



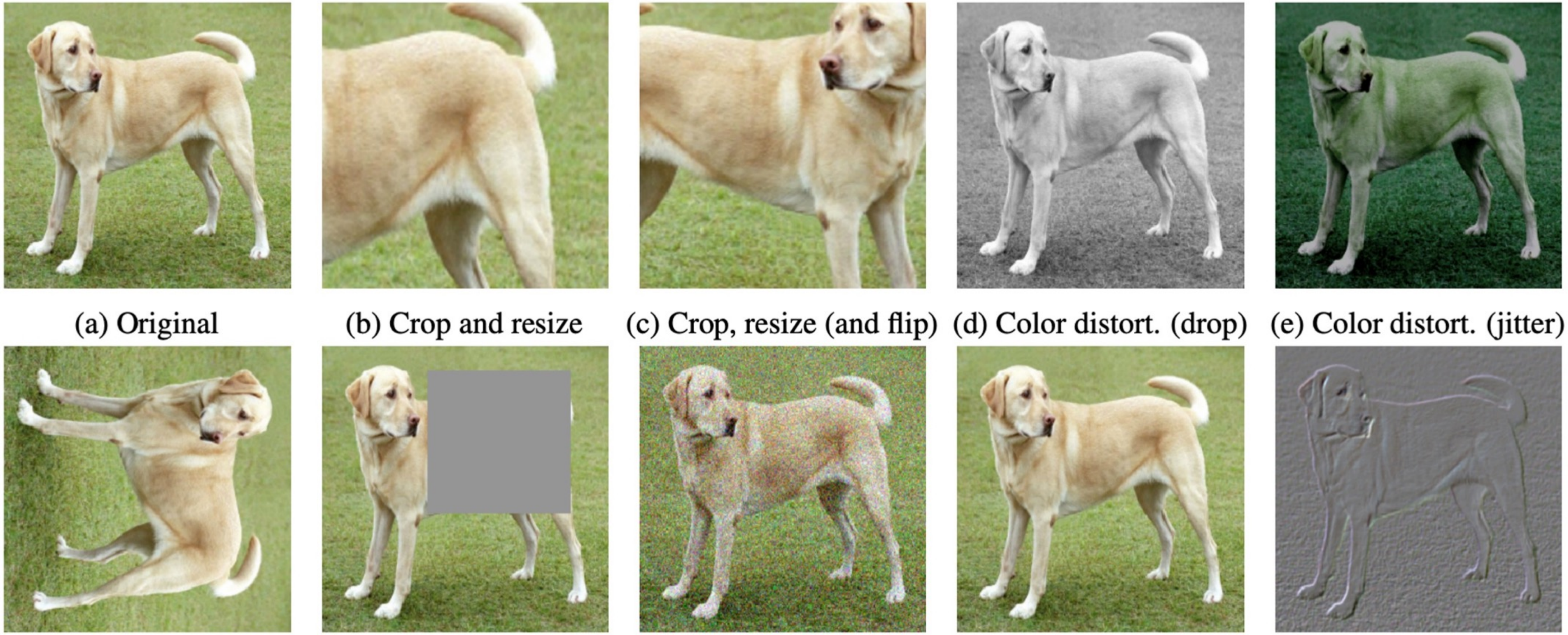
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

2. Attach a shallow network on the feature extractor; train the shallow network on the target task with small amount of labeled data

Contrastive Learning



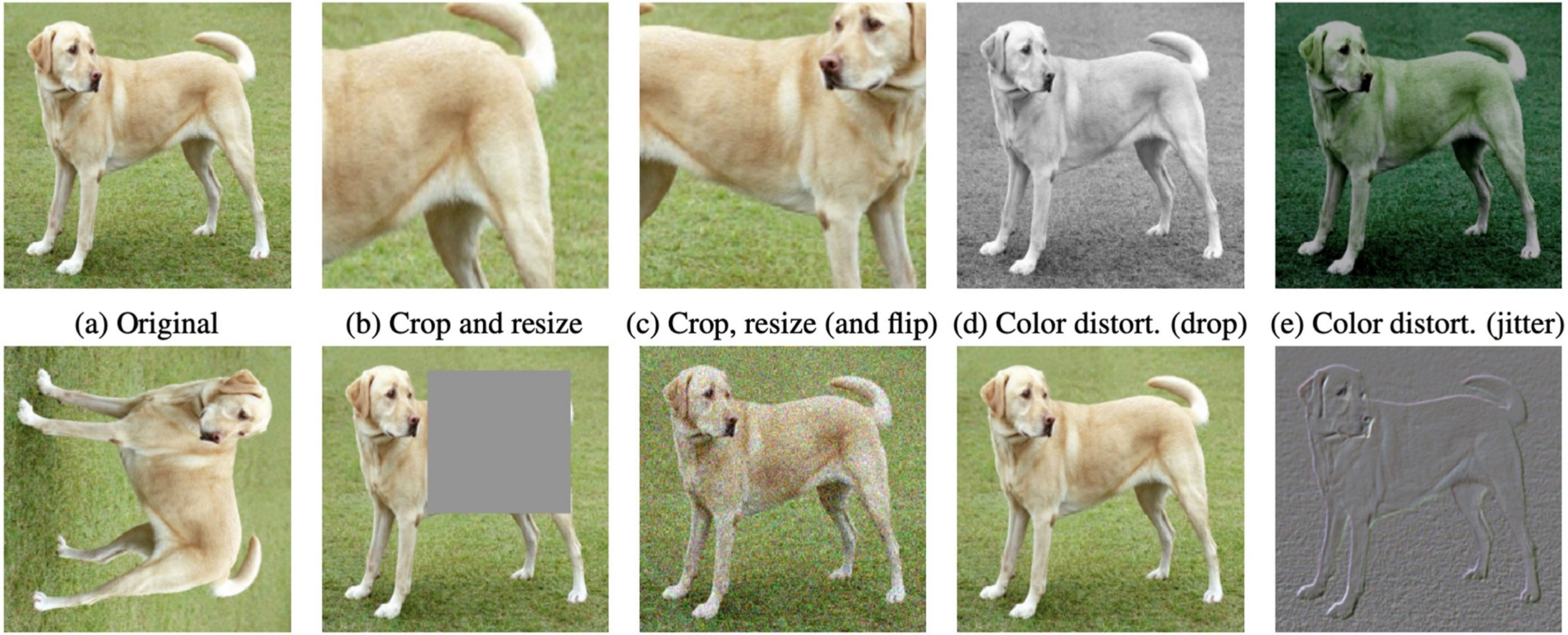
Representation Learning: SimCLR



(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

Representation Learning: SimCLR

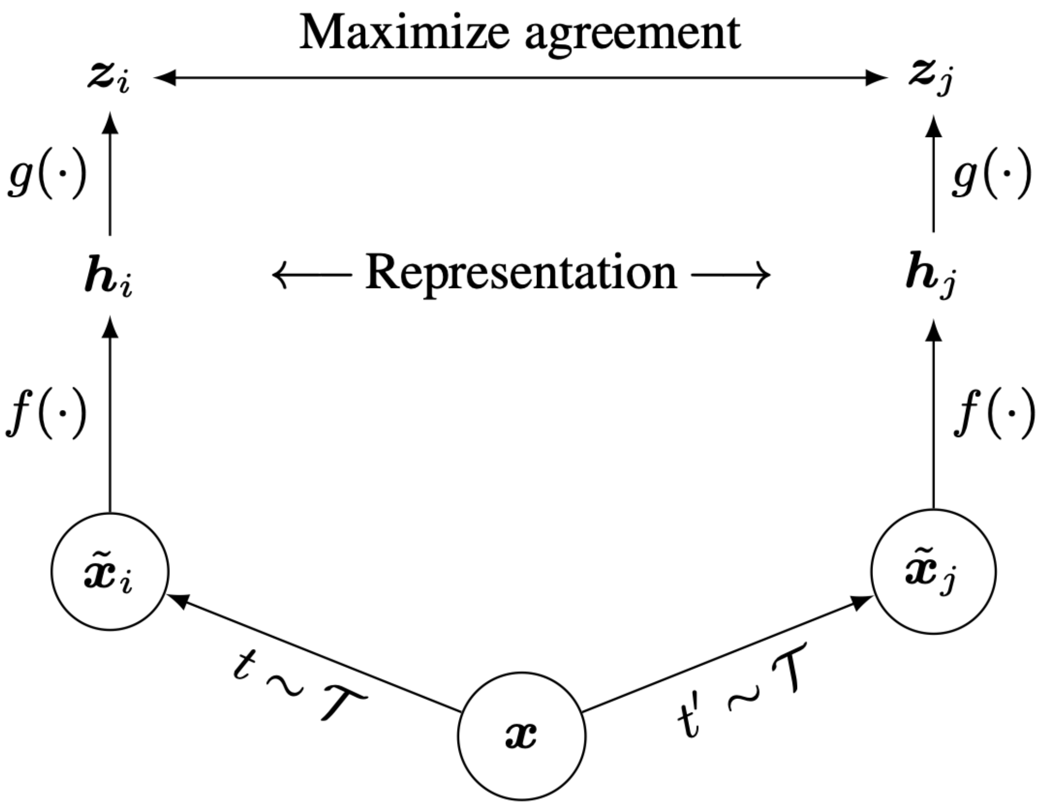


(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Representation Learning: SimCLR

Key idea: take N images, make 2N augmented versions, and then try to learn all the pairwise matchings

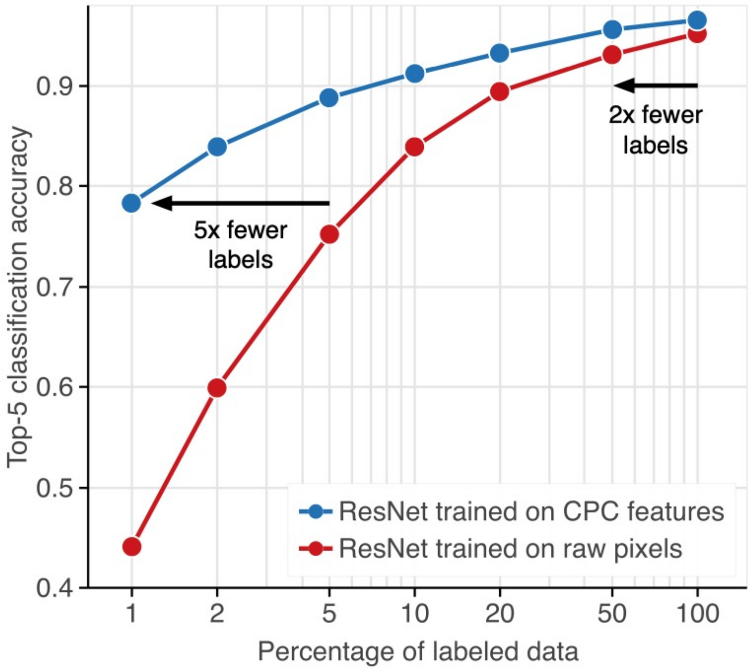


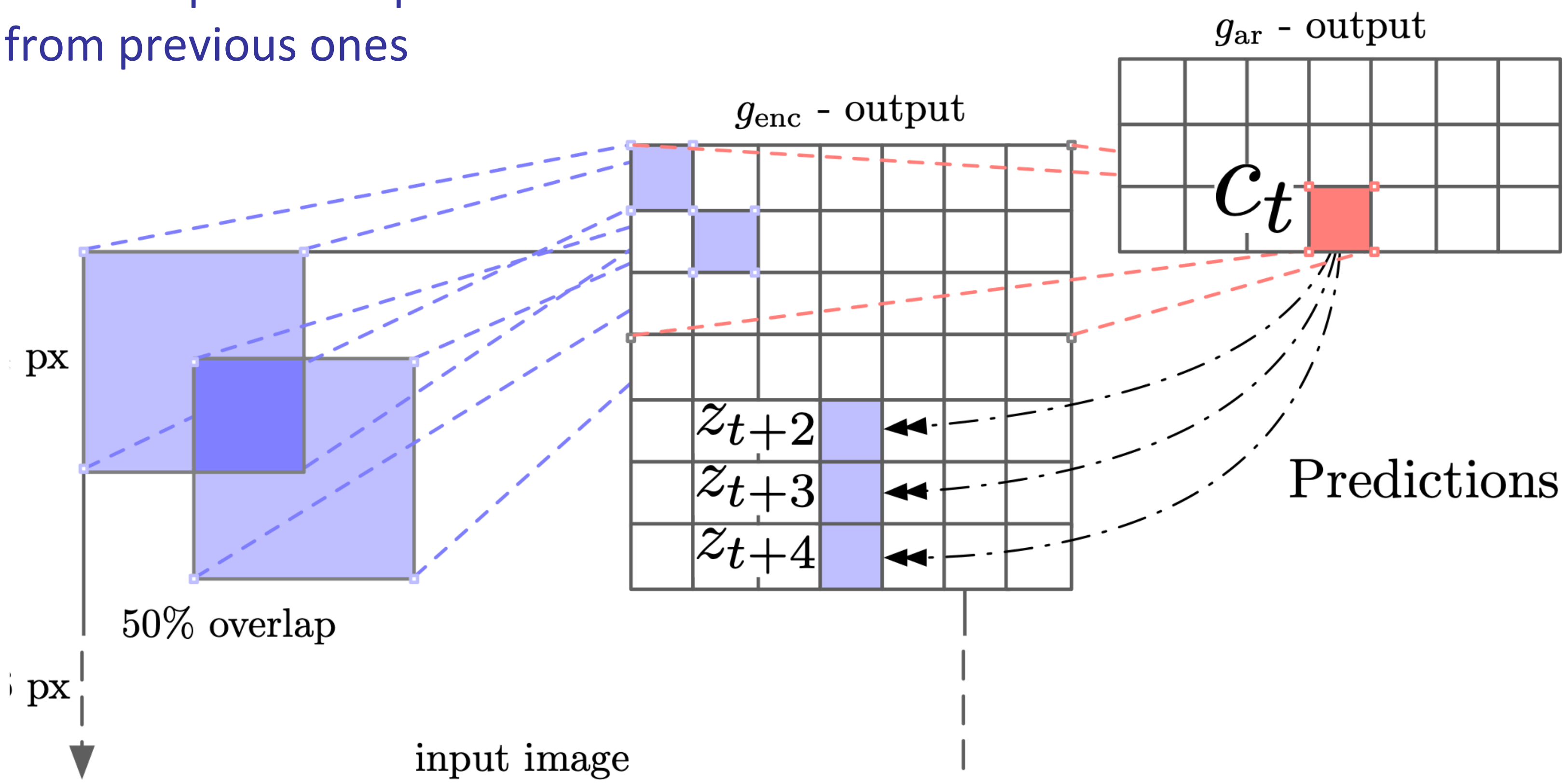
Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of f, g, $\mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i,j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

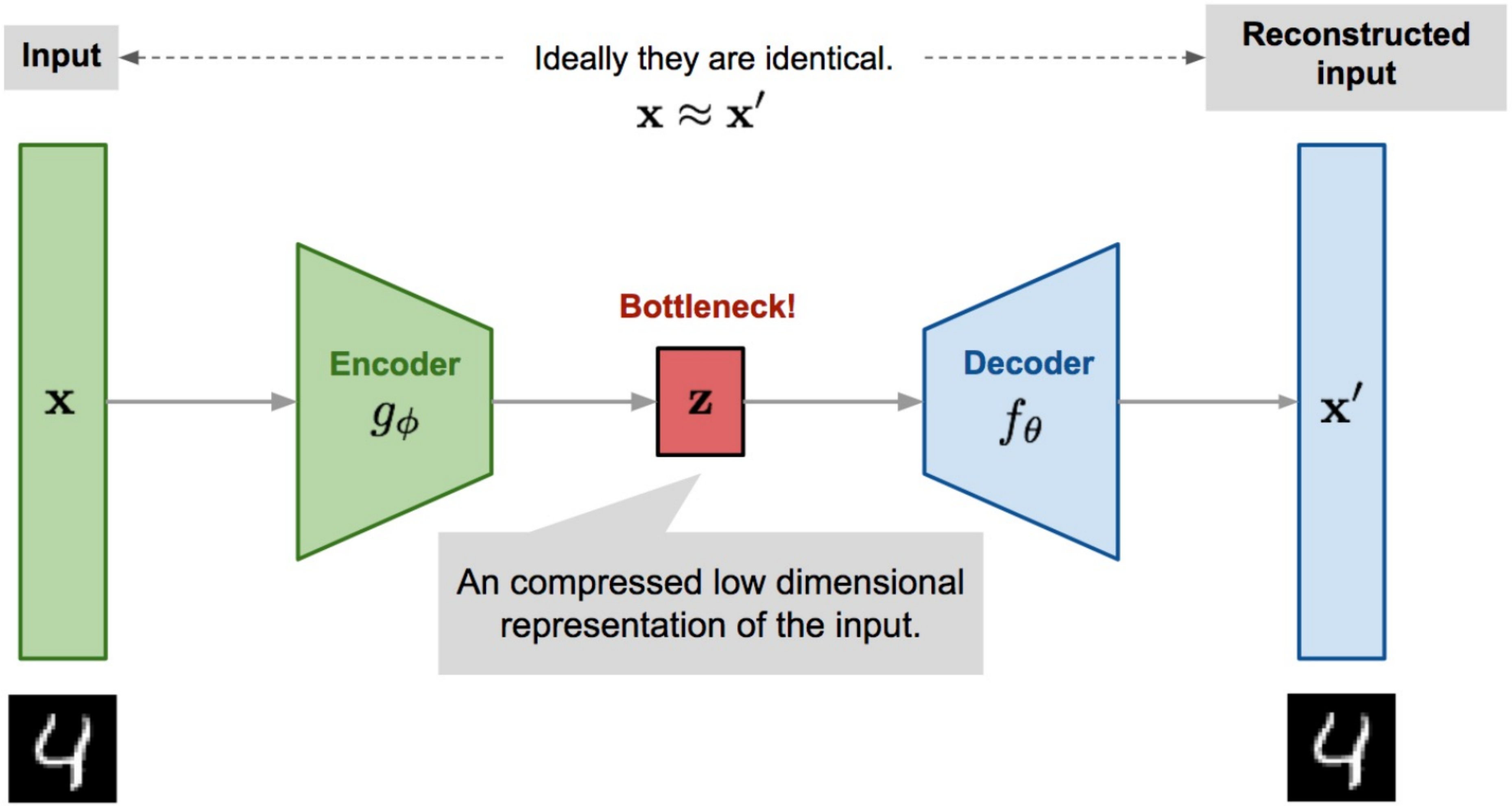
Representation Learning: CPC

Key idea: treat latent space in image as a sequence of patches
and learn to predict future patches from previous ones

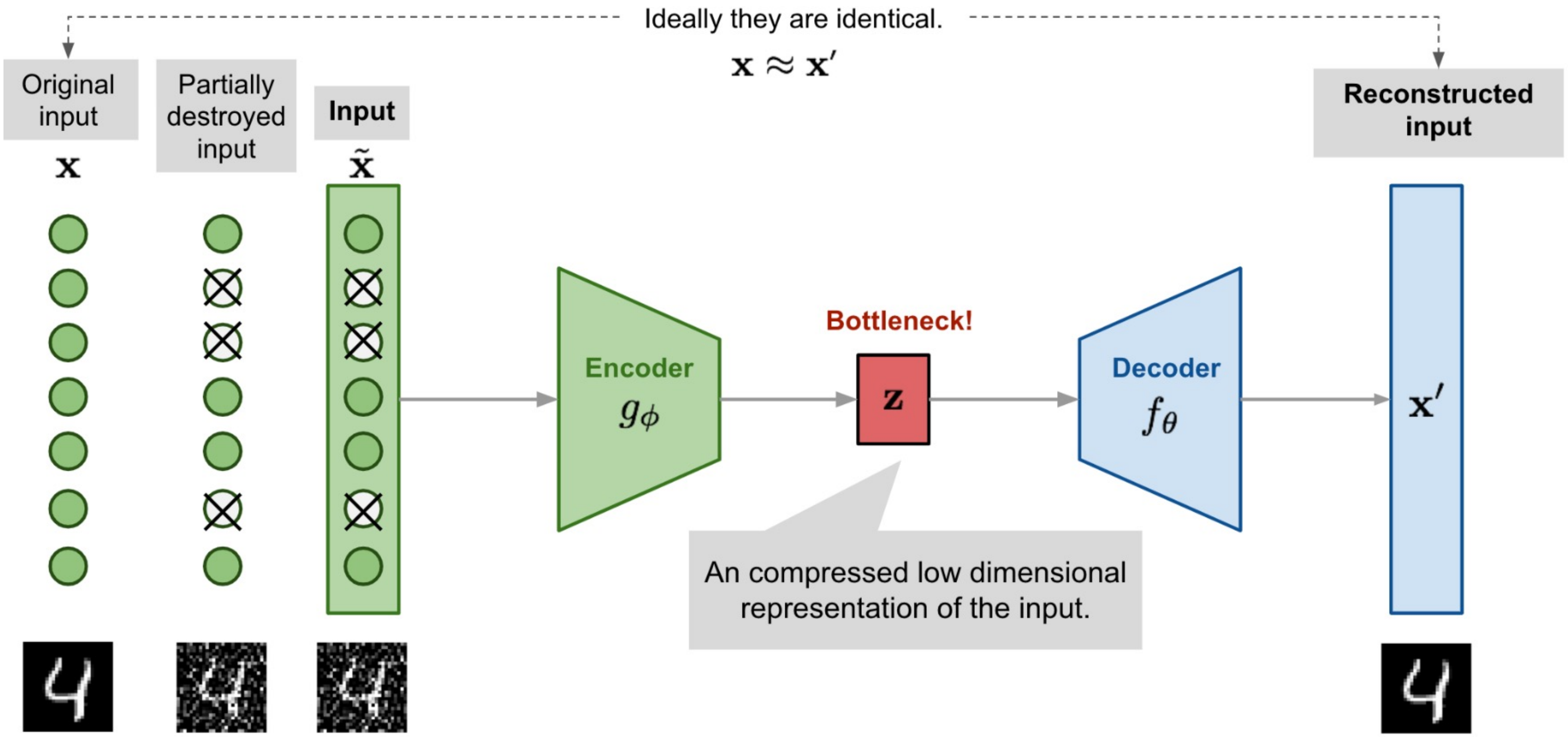




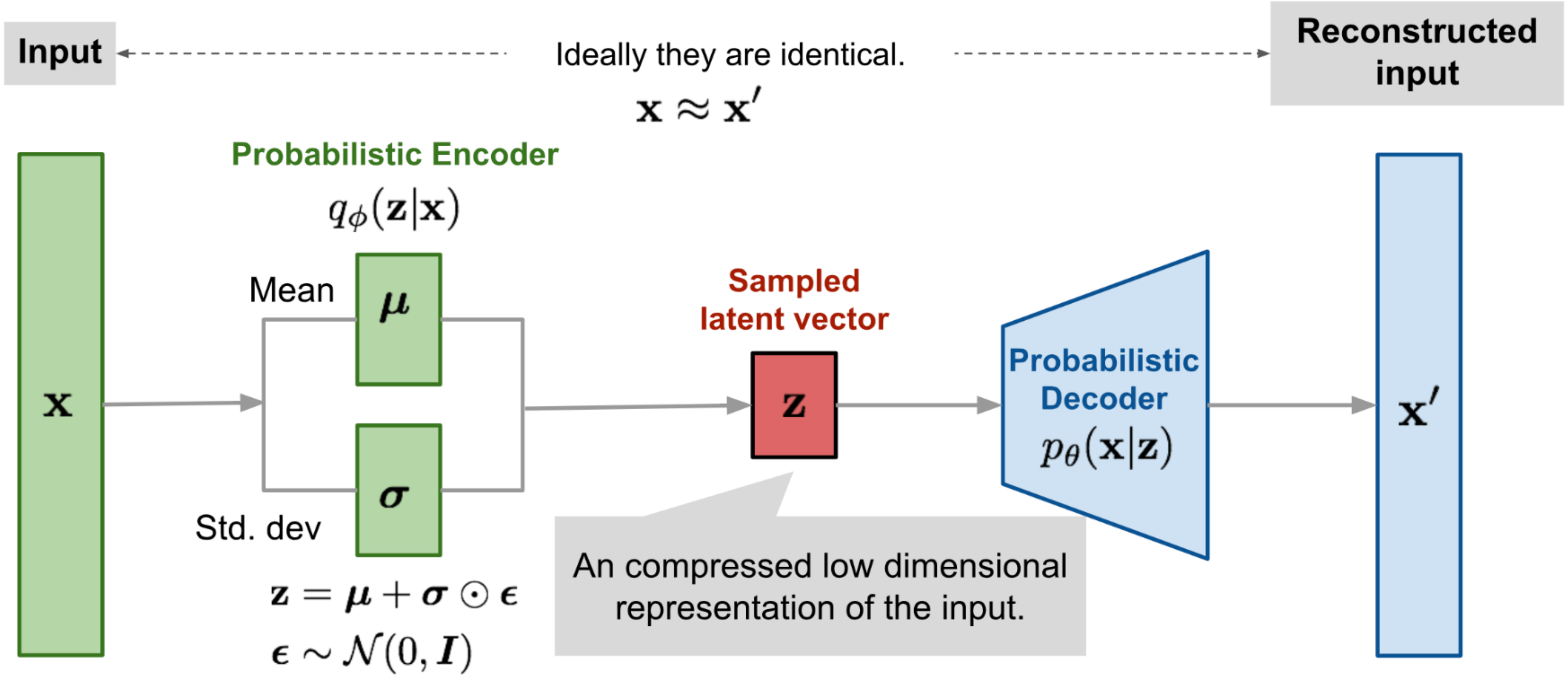
Autoencoders



Denoising Autoencoder



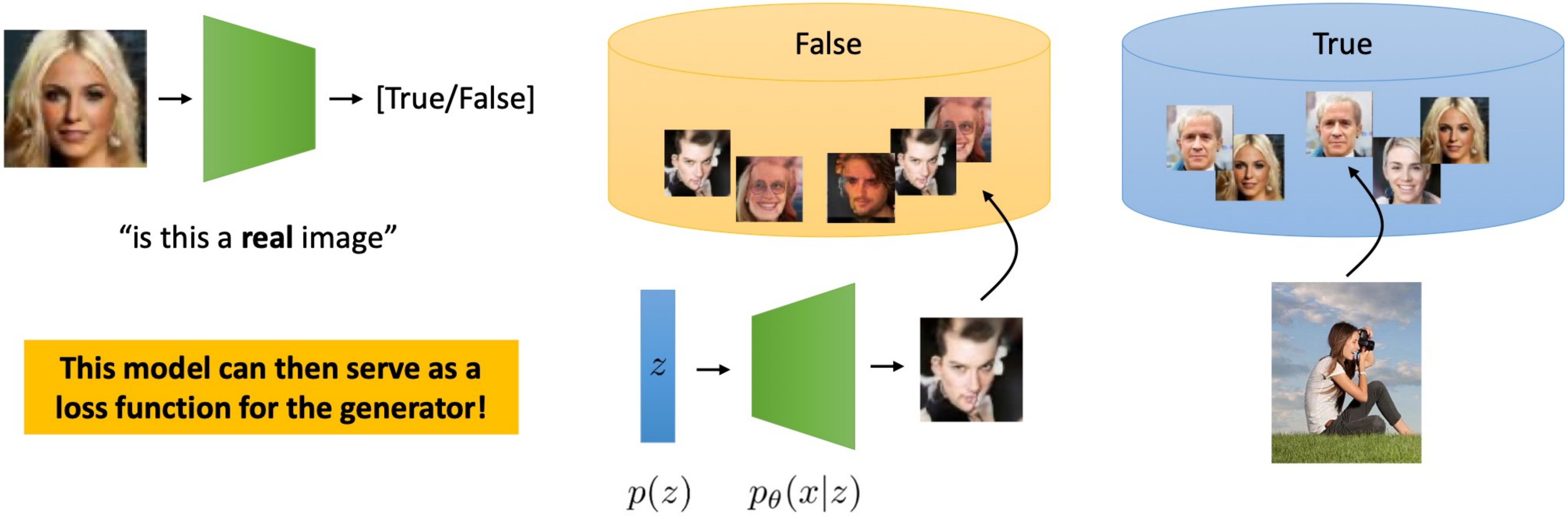
Variational Autoencoder



https://lilianweng.github.io/posts/2018-08-12-vae/

Generative Adversarial Networks

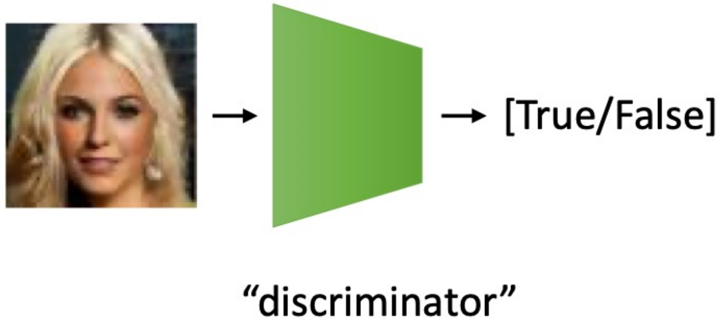
Idea: train a **network** to guess which images are real and which are fake!

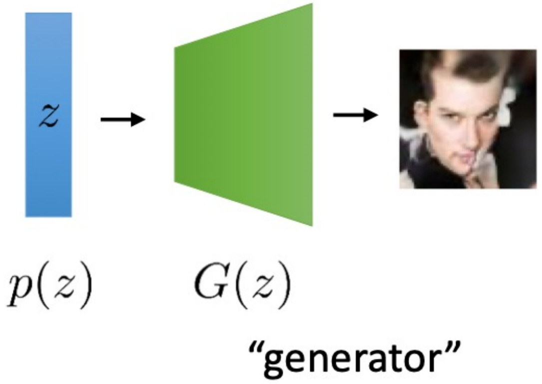


This model can then serve as a loss function for the generator!

Generative Adversarial Networks

1. get a "True" dataset $\mathcal{D}_T = \{(x_i)\}$
2. get a generator $G_\theta(z)$ random initialization!
3. sample a "False" dataset $\mathcal{D}_F$: $z \sim p(z)$, $x = G(z)$
4. update $D_\phi(x) = p_\phi(y|x)$ using $\mathcal{D}_T$ and $\mathcal{D}_F$ (1 SGD step)
5. use $D(x)$ to update $G(z)$ (1 SGD step)





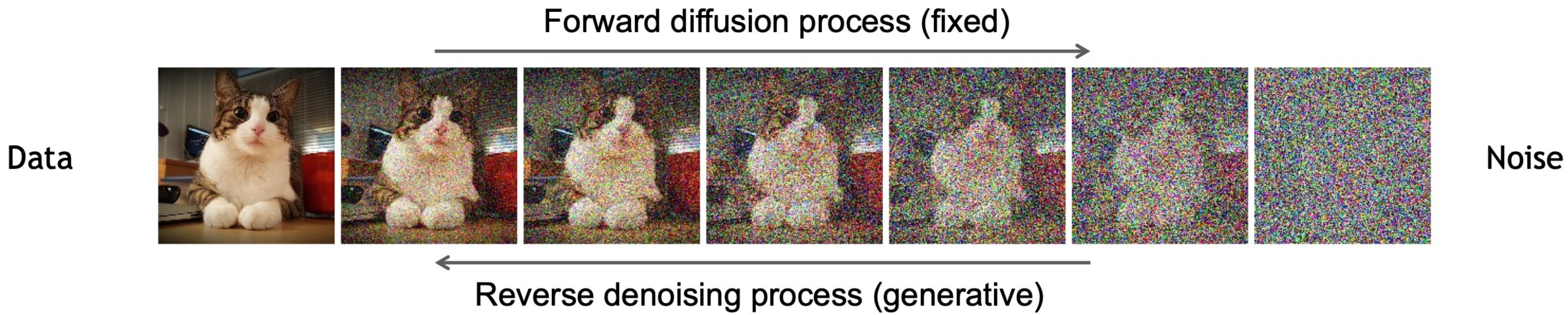
"classic" GAN 2-player game:

$$\min_G \max_D V(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \log D(x_i) \quad x_i \in \mathcal{D}_T \qquad \approx \frac{1}{N}\sum_{j=1}^{N} \log(1 - D(x_j))$$

$$x_j = G(z_j)$$

Diffusion Models



CLIP and DALL-E