Topics:

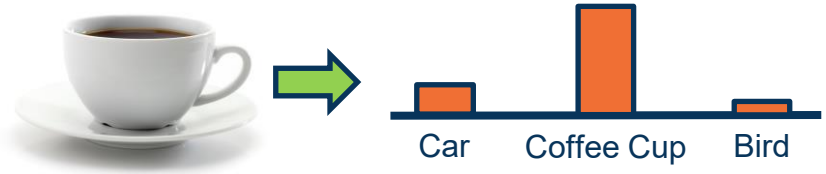
- Generative Adversarial Networks
- LLaVA-OneVision Discussion

CS 4644-DL / 7643-A
ZSOLT KIRA

- **Assignment 3**
  - Due **July 13th 11:59pm EST**
- **Projects**
  - Project proposal due **July 26th**
- Next Meta office hours 07/09 3pm ET on bias/fairness
  - NOT recorded!

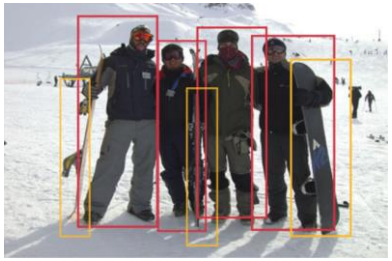
W9: July 7	Generative Models (Part I): Generative Adversarial Networks READING: LLaVA-OneVision
W9: July 9	Generative Models (Part II): Diffusion Models PS3/HW3 due July 13th 11:59pm (grace period July 15th)
W10: July 14	Variational Autoencoders (VAEs)
W10: July 16	Open topic! (please suggest). Otherwise default is Object Detection and Segmentation
W11: July 21	Fairness/Bias Discussion, open topics not covered, Wrap-up Final Project Due July 26th 11:59pm (grace period July 28th)

Image to Image CNNs



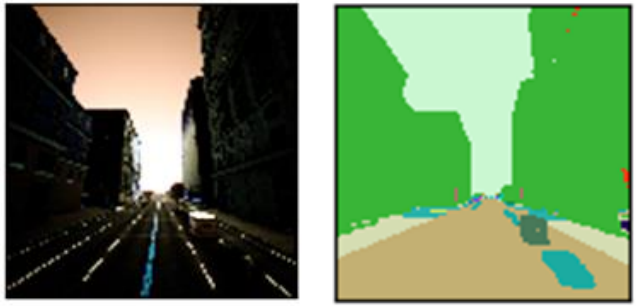
Classification

(Class distribution per image)



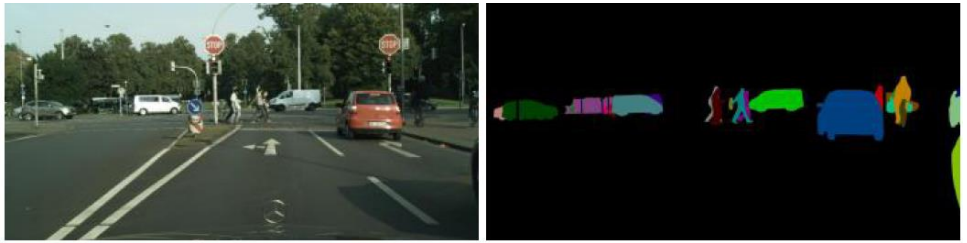
Object Detection

(List of bounding boxes with class distribution per box)



Semantic Segmentation

(Class distribution per pixel)

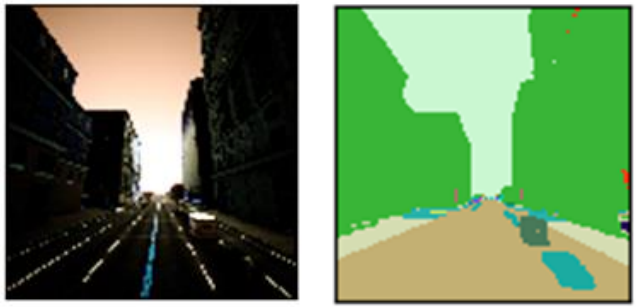


Instance Segmentation

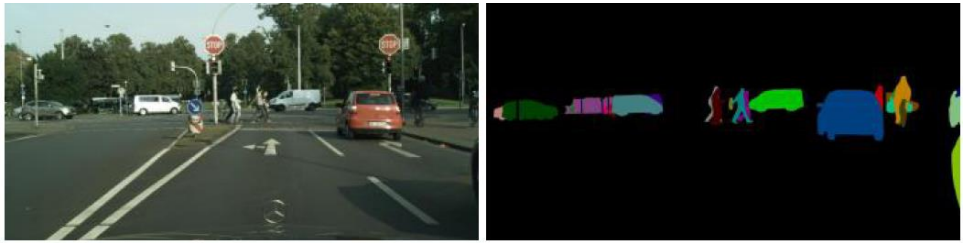
(Class distribution per pixel with unique ID)

Given an image, output another image

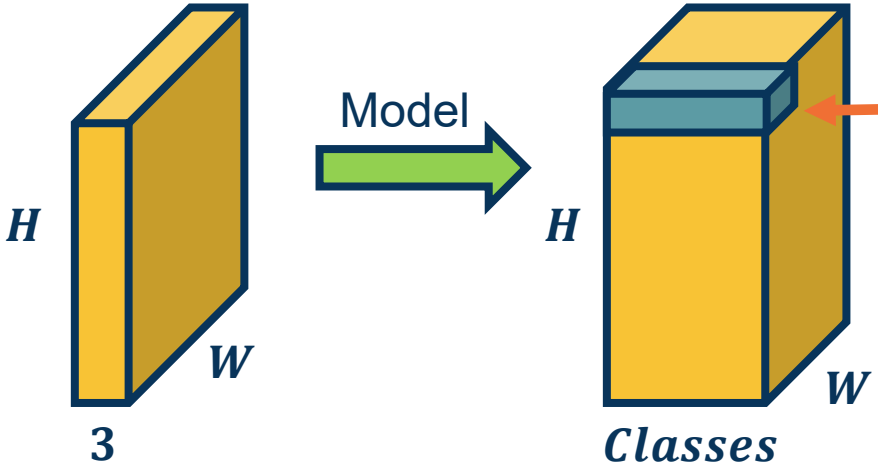
- Each output contains class distribution per pixel
- More generally an image-to-image problem

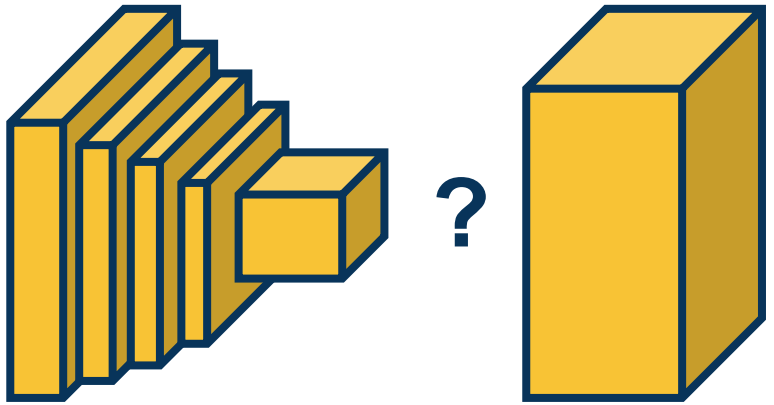


Semantic Segmentation
(Class distribution per pixel)

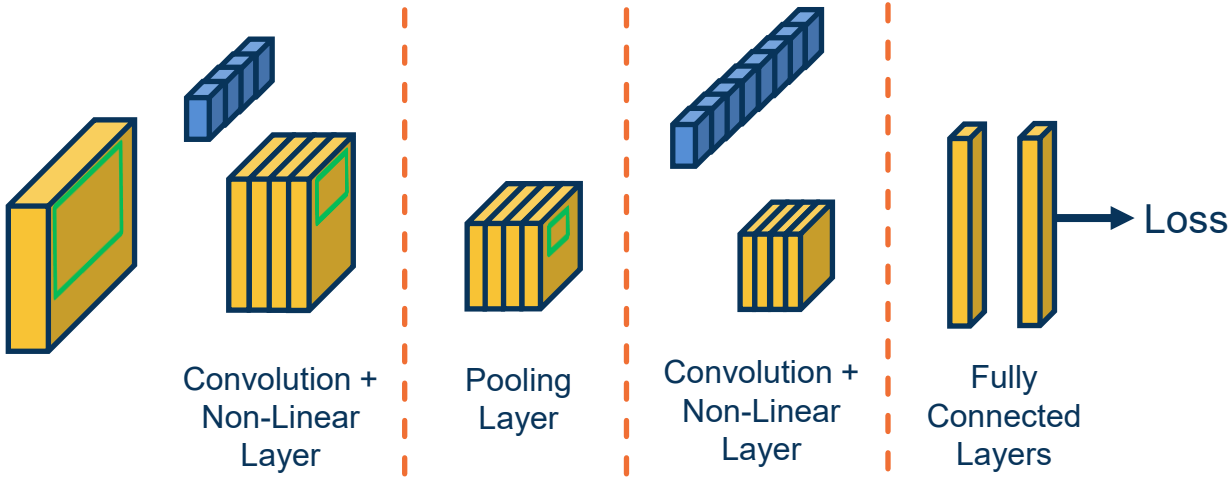


Instance Segmentation
(Class distribution per pixel with unique ID)





Input & Output

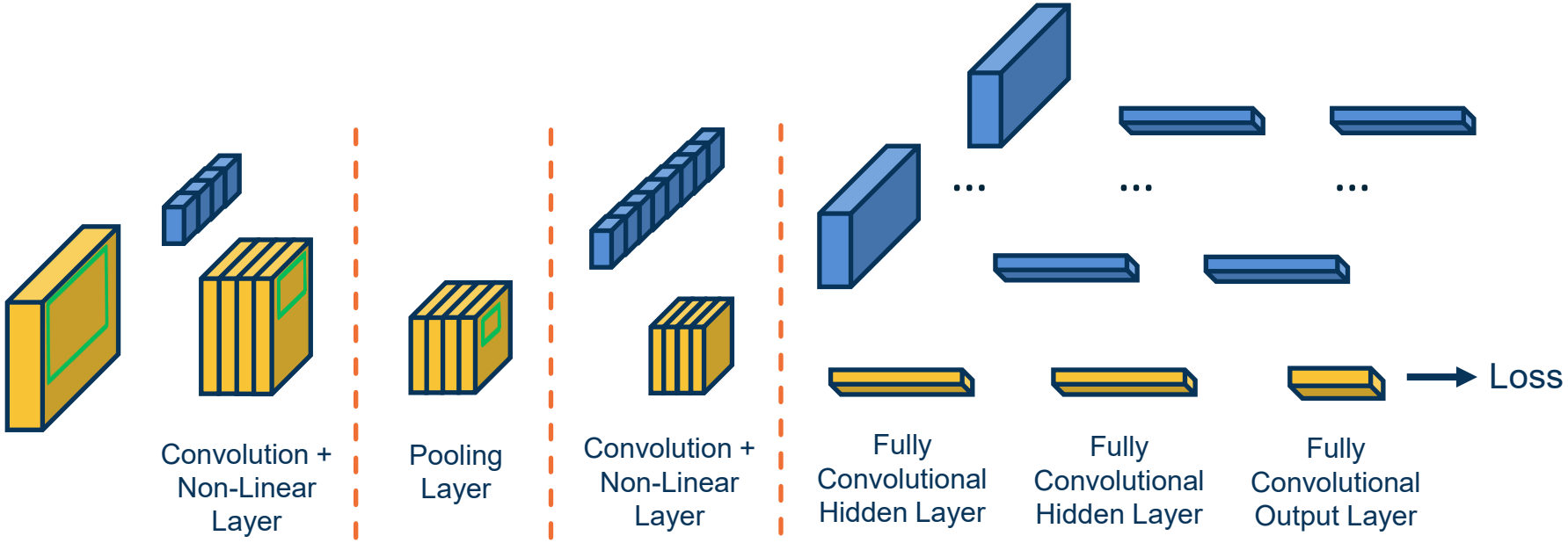


Fully connected layers no longer explicitly retain spatial information (though the network can still learn to do so)

Idea: Convert fully connected layer to convolution!

Idea 1: Fully-Convolutional Network

Georgia Tech



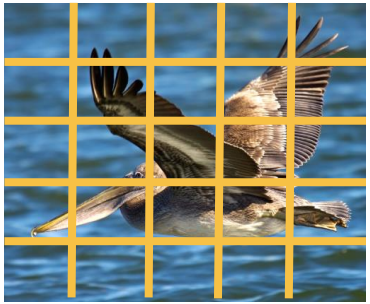
Each kernel has the size of entire input! (output is 1 scalar)

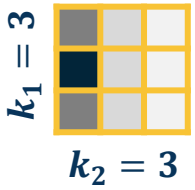
- This is equivalent to Wx+b!
- We have one kernel per output node

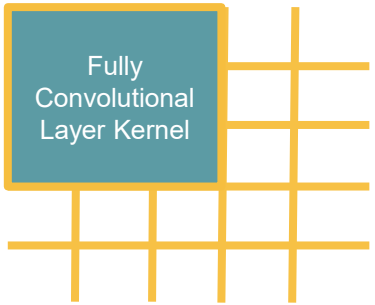
Converting FC Layers to Conv Layers

Georgia Tech

Original:

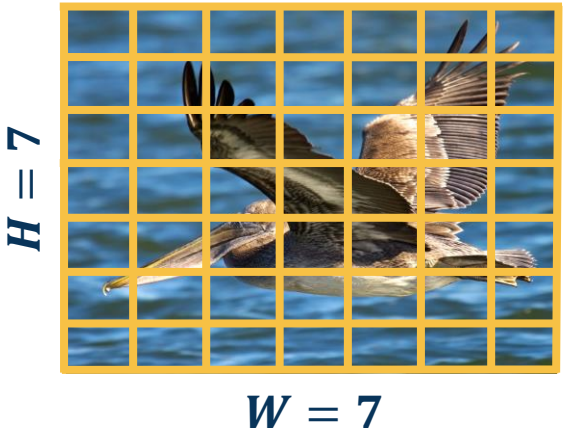


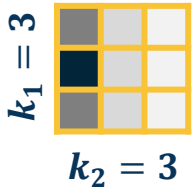


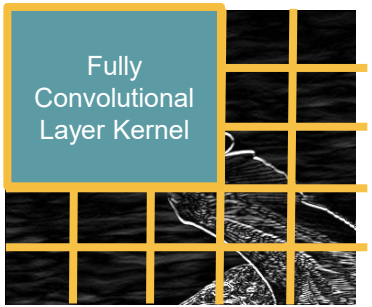


Input **Conv Kernel** **Output**

Larger:







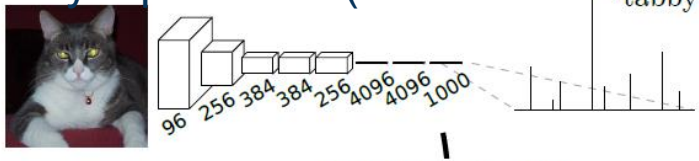
Same Kernel, Larger Input

Georgia Tech

Why does this matter?

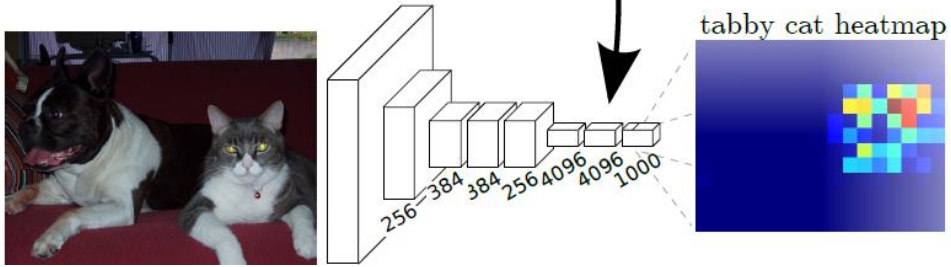
- We can stride the “fully connected” classifier across larger inputs!
- Convolutions work on arbitrary input sizes (because of striding)

Original sized image



convolutionalization

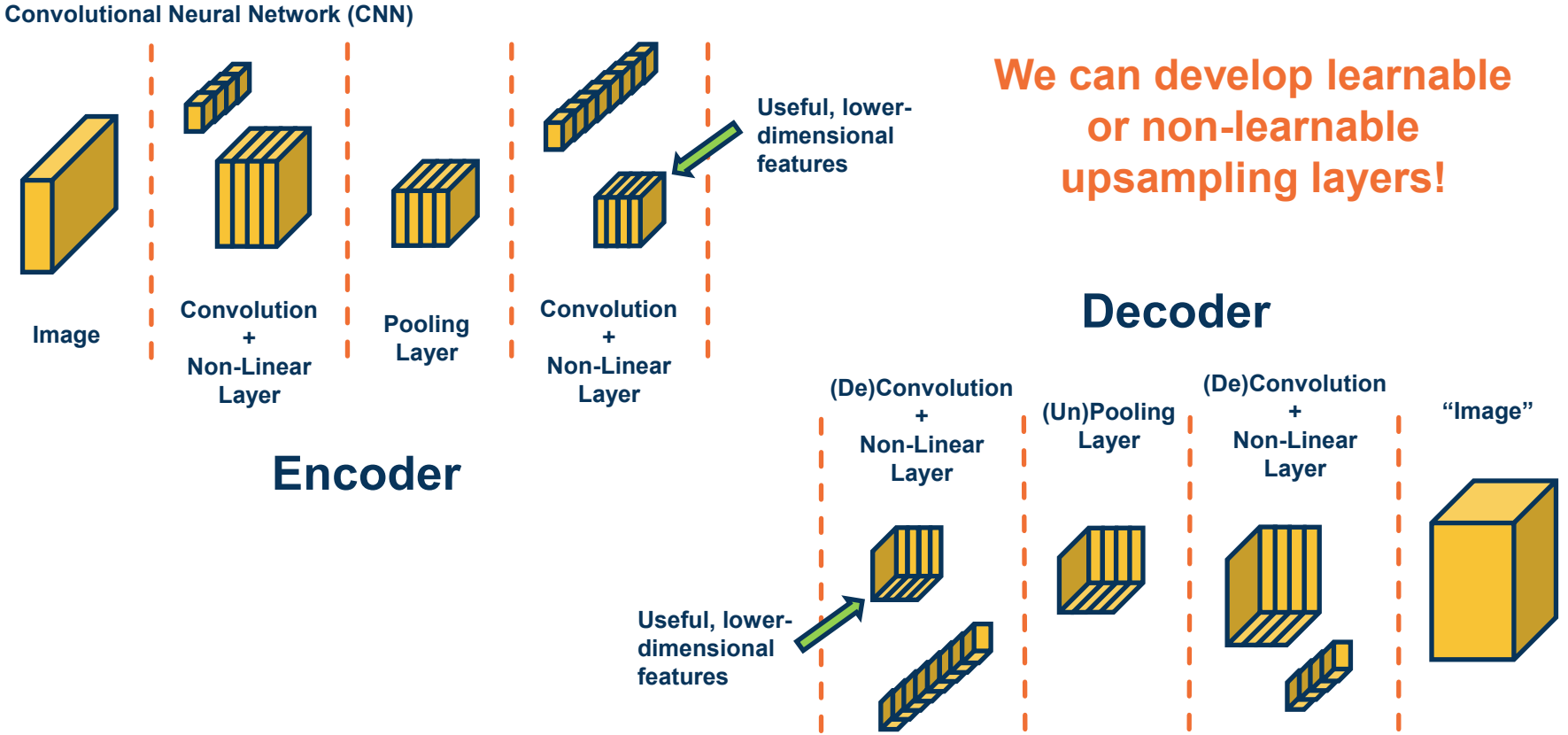
Larger Image



Larger Output Size!

Larger Output Maps

Long, et al., “Fully Convolutional Networks for Semantic Segmentation”, 2015

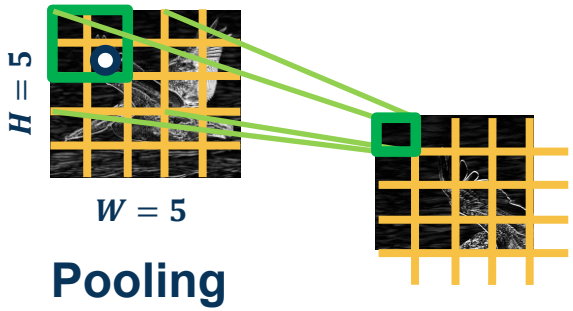


Idea 2: "De"Convolution and UnPooling

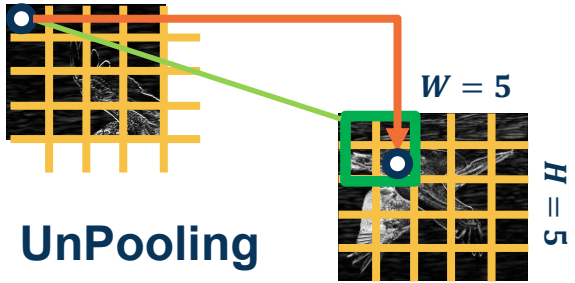
Example : Max pooling

- Stride window across image but perform per-patch **max operation**

$$\mathbf{X}(0:1, 0:1) = \begin{bmatrix} 100 & 150 \\ 100 & 200 \end{bmatrix} \Rightarrow \max(0:1,0:1) = 200$$



Copy value to position chosen as max in encoder, fill reset of this window with zeros



Idea: Remember max elements in encoder! Copy value from equivalent position, rest are zeros

$$X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix}$$

2x2 max pool

$$Y = \begin{bmatrix} 150 & 150 \\ 100 & 110 \end{bmatrix}$$

Encoder

Decoder

$$X = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix}$$

2x2 max unpool

$$Y = \begin{bmatrix} 0 & 300 & - \\ 0 & 0 & - \\ - & - & - \end{bmatrix}$$

Max Unpooling Example (one window)

Georgia Tech

$$X_{enc} = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix}$$

2x2 max pool

$$Y_{enc} = \begin{bmatrix} 150 & 150 \\ 100 & 110 \end{bmatrix}$$

Encoder

Contributions from multiple windows are summed

Decoder

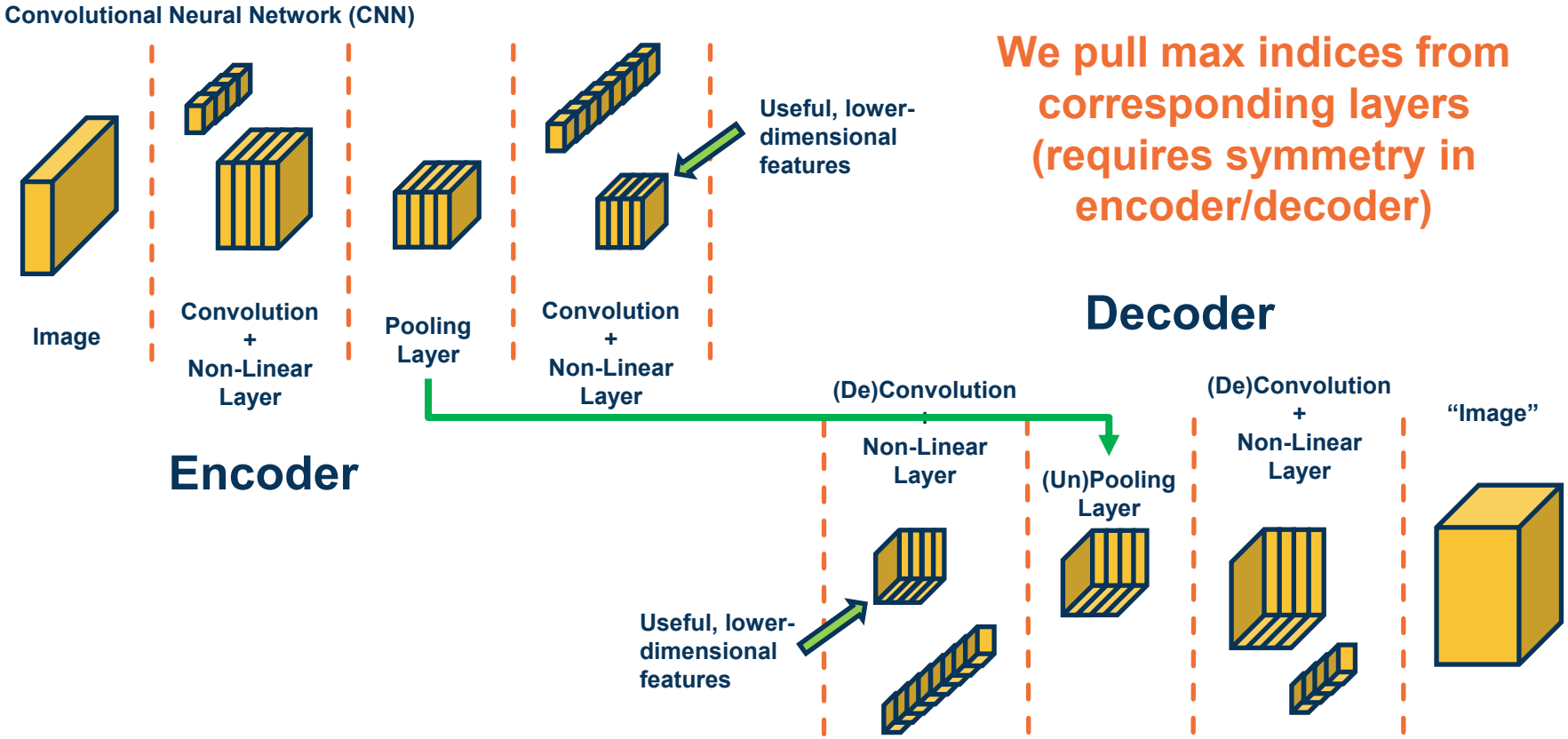
$$X_{dec} = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix}$$

2x2 max unpool

$$Y_{dec} = \begin{bmatrix} 0 & 300+450 & 0 \\ 100 & 0 & 250 \\ 0 & 0 & 0 \end{bmatrix}$$

Max Unpooling Example

Georgia Tech

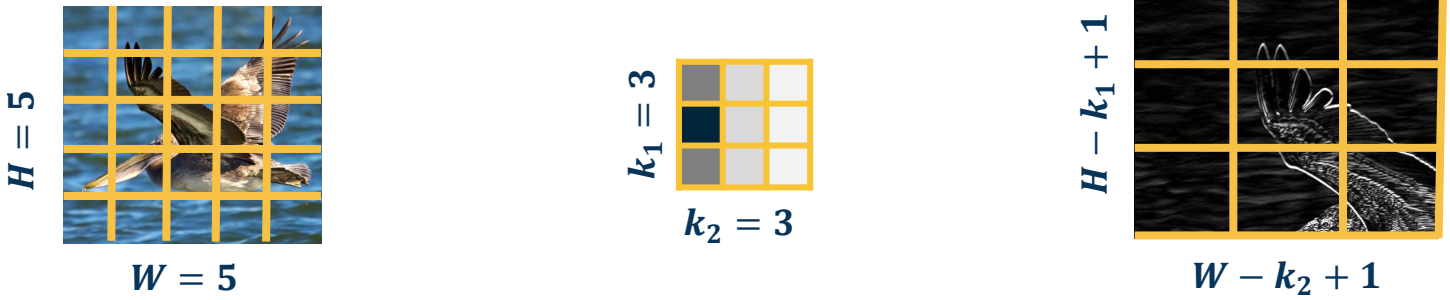


Symmetry in Encoder/Decoder

Georgia Tech

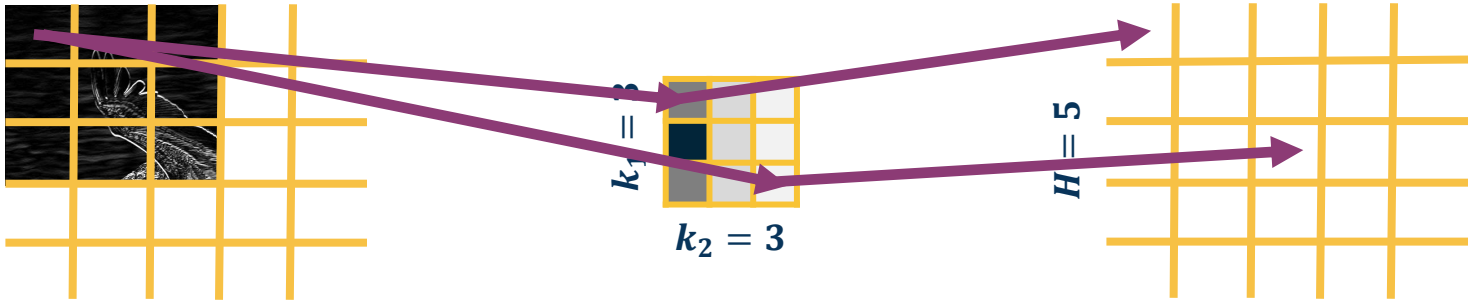
How can we *upsample* using convolutions and learnable kernel?

Normal Convolution



Transposed Convolution (also known as "deconvolution", fractionally strided conv)

Idea: Take each input pixel, multiply by learnable kernel, "stamp" it on output



$$X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}$$

Contributions from multiple windows are summed

$$\begin{bmatrix} 120 & -120 & 0 & 0 \\ 240 & -240 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

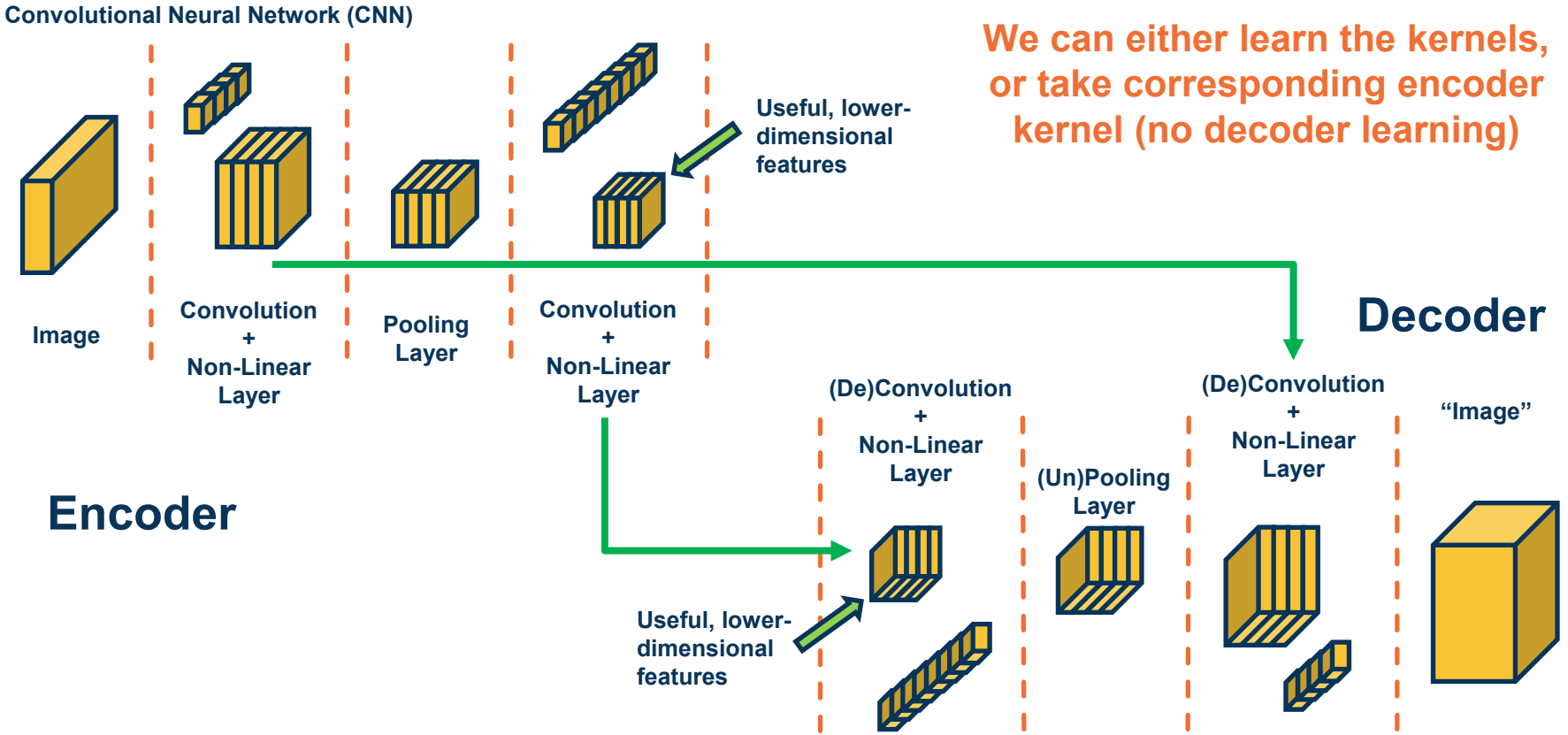
Incorporate
X(0,0)

$$\begin{bmatrix} 120 & -120+150 & -150 & 0 \\ 240 & -240+300 & -300 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Incorporate
X(1,0)

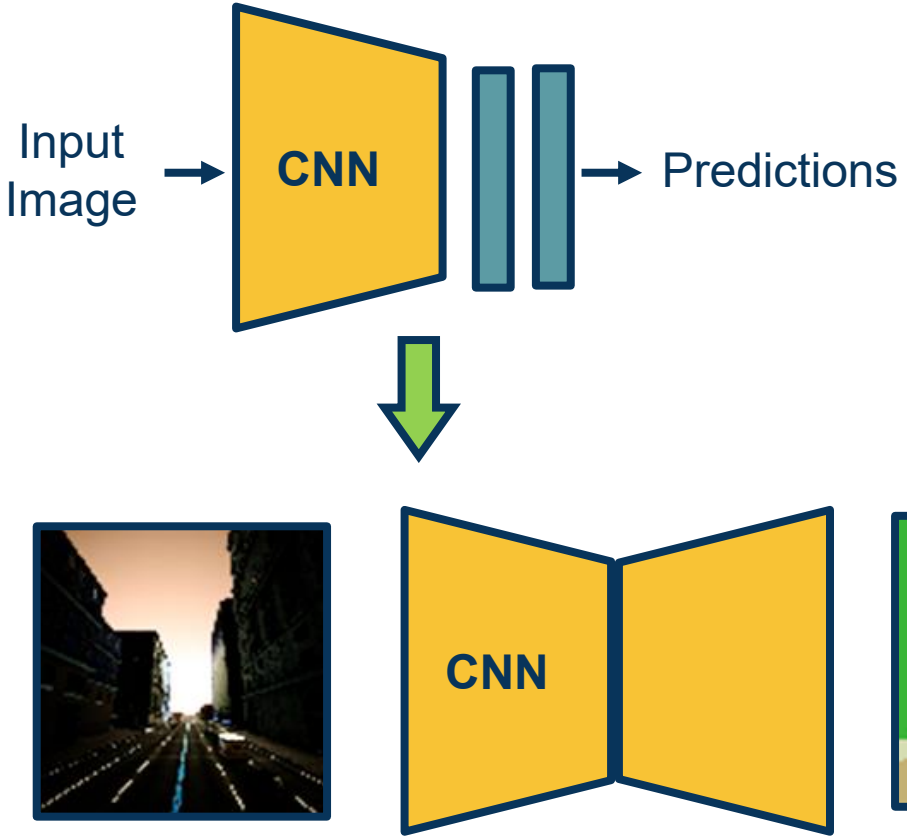
Transposed Convolution Example

Georgia Tech



Symmetry in Encoder/Decoder

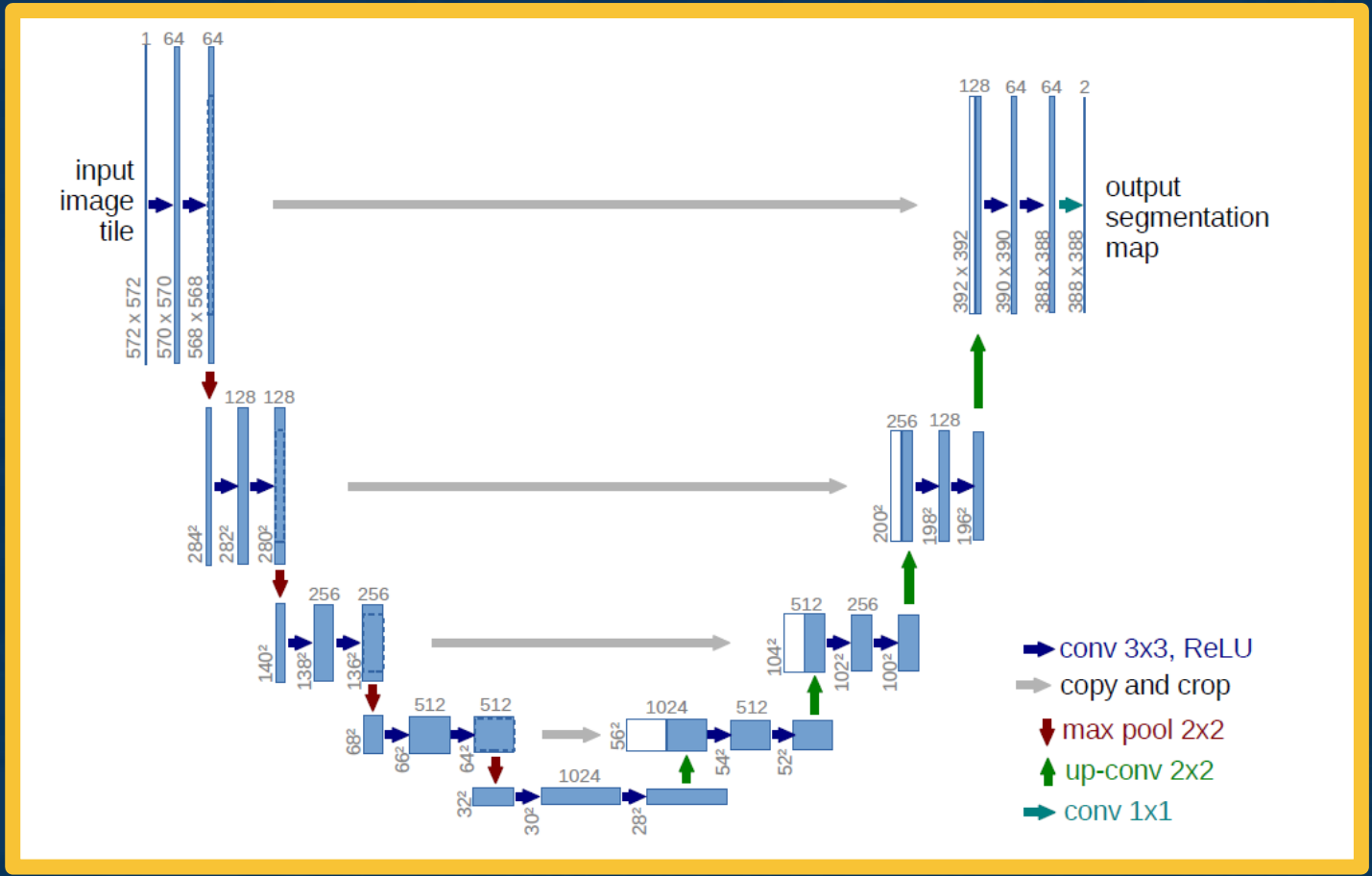
Georgia Tech



We can start with a pre-trained trunk/backbone (e.g. network pretrained on ImageNet)!

U-Net

You can have skip connections to bypass bottleneck!



Ronneberger, et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

Georgia Tech

Summary

- Various ways to get **image-like outputs,** for example to predict segmentations of input images
- Fully convolutional layers essentially apply the striding idea to the output classifiers, supporting arbitrary input sizes
  - (without output size depending on what the input size is)
- We can have various upsampling layers that actually increase the size
- Encoder/decoder architectures are popular ways to leverage these to perform general image-to-image tasks

Georgia Tech

Generative Models: Introduction

Supervised Learning

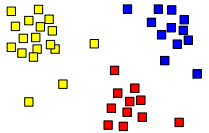
- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, $P(y|x)$
- e.g. classification

Sheep
Dog
Cat
Lion
Giraffe

Less Labels

Unsupervised Learning

- Input: $\{X\}$
- Learning output: $P(x)$
- Example: Clustering, density estimation, etc.

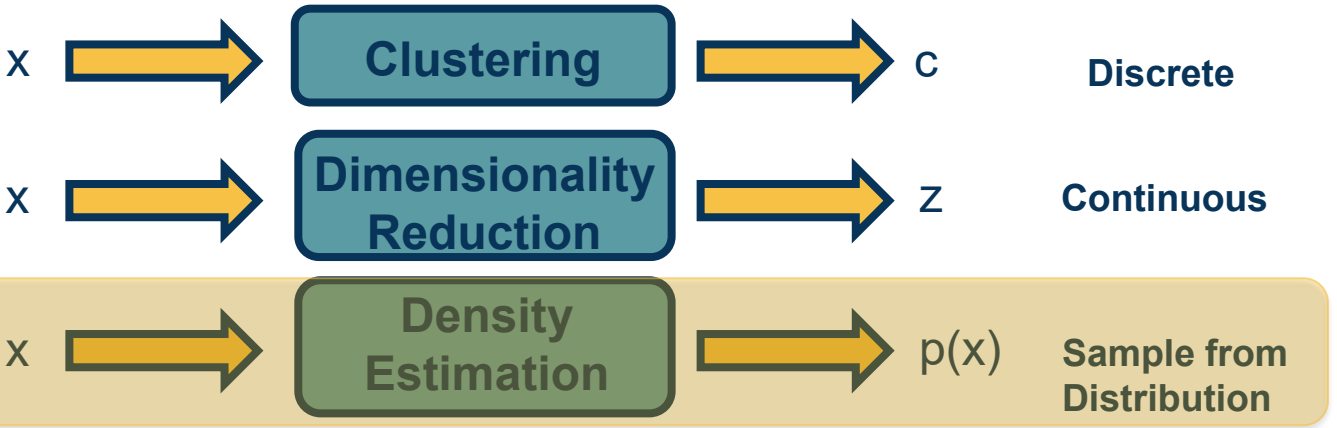


Spectrum of Low-Labeled Learning

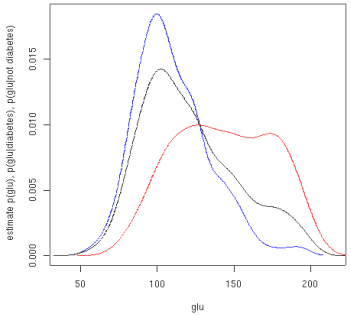
Georgia Tech

Supervised Learning

Unsupervised Learning



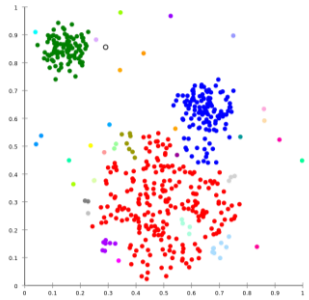
Traditional unsupervised learning methods:



Density estimation

Modeling $P(x)$

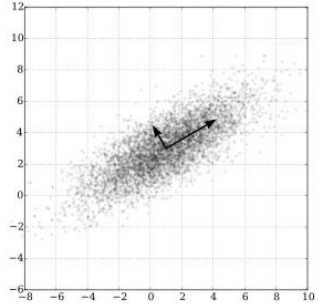
Deep Generative Models



Clustering

Comparing/ Grouping

Metric learning & clustering



Principal Component Analysis

Representation Learning

Almost all deep learning!

Similar in deep learning, but **from neural network/learning perspective**

Discriminative vs. Generative Models

- Discriminative models model $P(y|x)$
  - Example: Model this via neural network, SVM, etc.

- Generative models model $P(x)$

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Discriminative vs. Generative Models

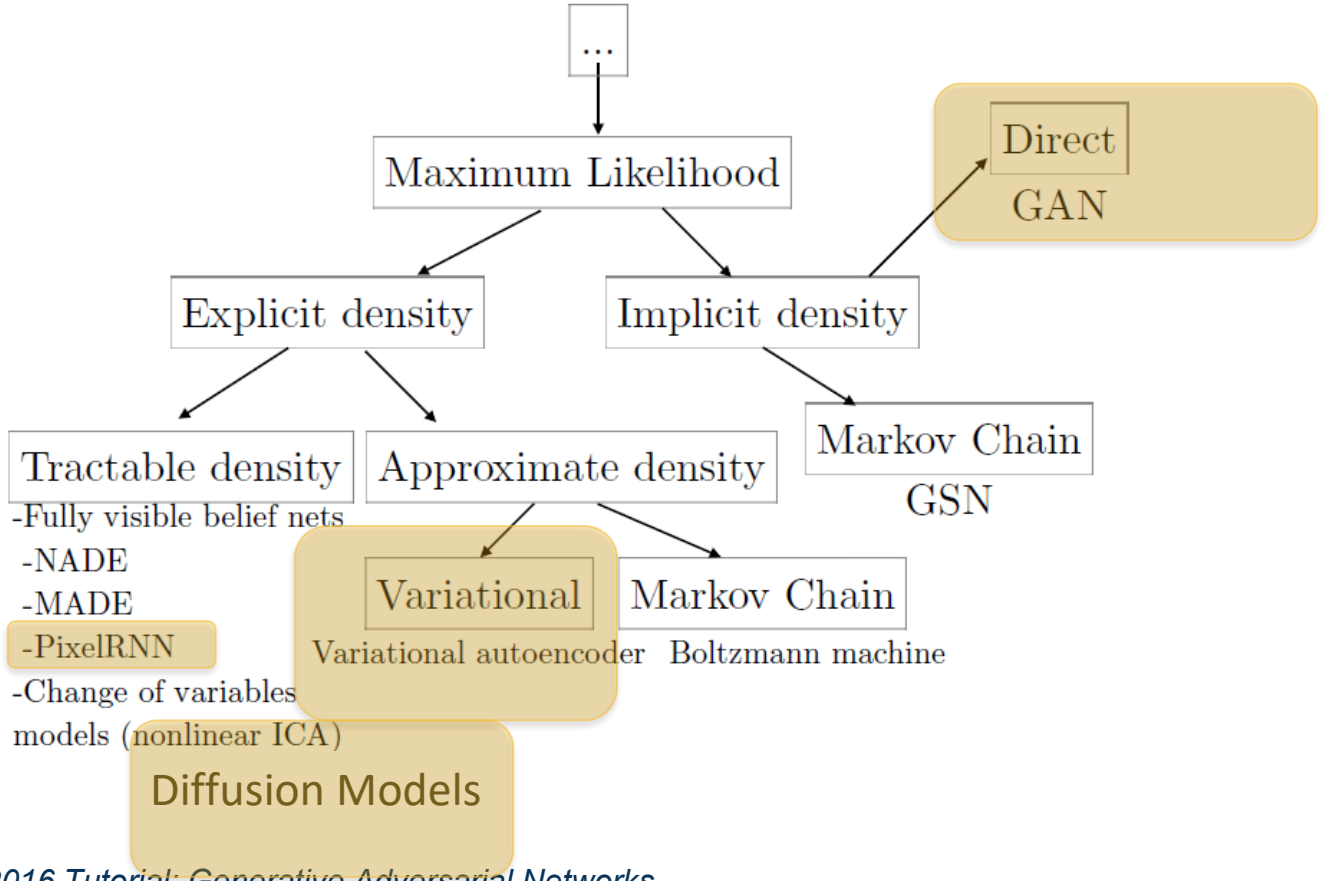
- Discriminative models model $P(y|x)$
  - Example: Model this via neural network, SVM, etc.

- Generative models model $P(x)$
- We can parameterize our model as $P(x, \theta)$ and use maximum likelihood to optimize the parameters given an unlabeled dataset:

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \\ &= \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{m} p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) \\ &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\text{model}}\left(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right).\end{aligned}$$

- They are called generative because they can often generate *samples*
  - Example: Multivariate Gaussian with estimated parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$

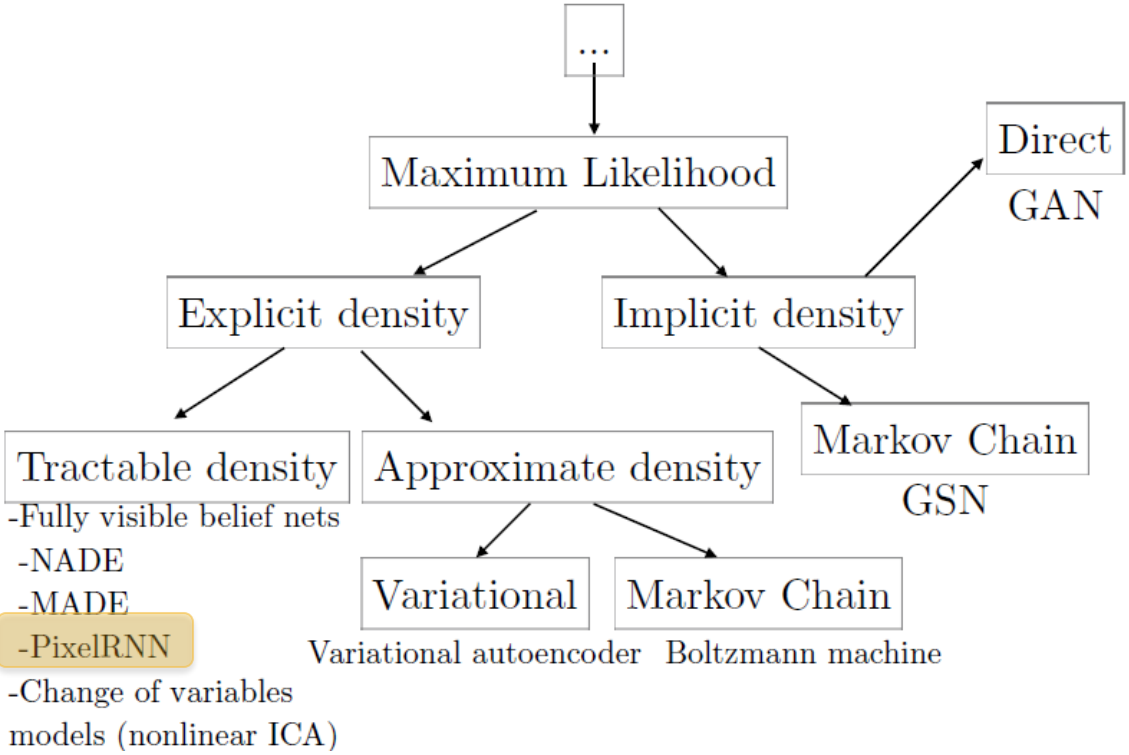
Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

PixelRNN & PixelCNN



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

We can use chain rule to decompose the joint distribution

- Factorizes joint distribution into a product of conditional distributions
  - Similar to Bayesian Network (factorizing a joint distribution)
  - Similar to language models!

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

Same as language modeling!

$$p(s) = \prod_{i} p(w_i \mid w_{i-1}, \ldots, w_1)$$

i: next word; $w_{i-1}, \ldots, w_1$: history

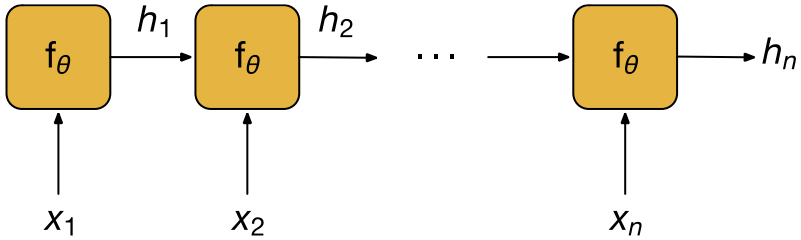
- Requires some *ordering* of variables (edges in a probabilistic graphical model)
- We can estimate this conditional distribution as a neural network

Oord et al., Pixel Recurrent Neural Networks

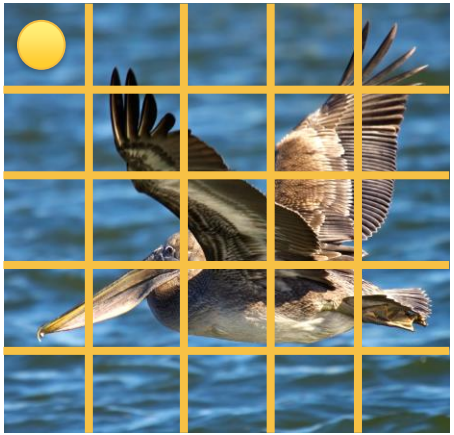
- Language modeling involves estimating a probability distribution over sequences of words.

$$\mathrm{p}(\boldsymbol{s}) = \mathrm{p}(w_1, w_2, \ldots, w_n) = \prod_i \mathrm{p}(\underbrace{w_i}_{\substack{\text{next} \\ \text{word}}} \mid \underbrace{w_{i-1}, \ldots, w_1}_{\text{history}})$$

- RNNs are a family of neural architectures for modeling sequences.



Language Models as an RNN



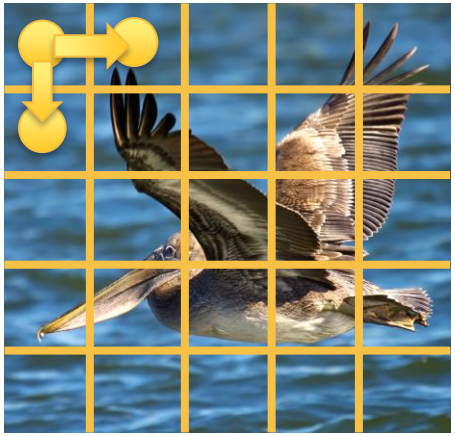
$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

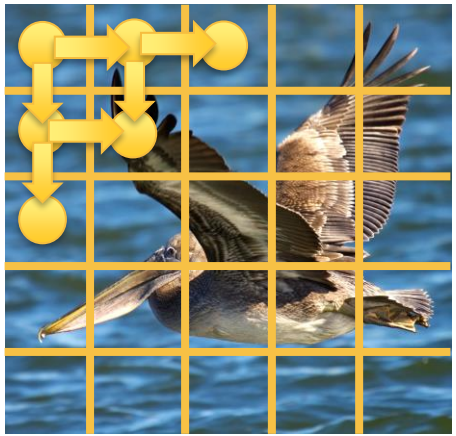
$$p(x) = p(x_1) \prod_{i=2}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

1. Choose ordering (upper left, top to bottom, left to right.

Separate out pixel 1

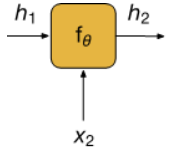
Oord et al., Pixel Recurrent Neural Networks





$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1)\prod_{i=1}^{n^2} p(x_i|x_1, \ldots, x_{i-1})$$

- Model this as RNN with parameters

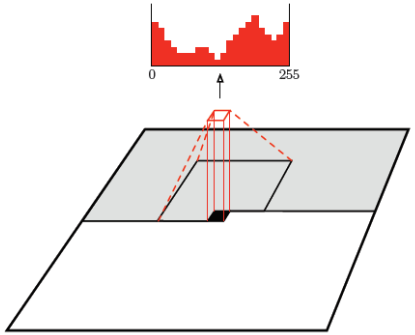


- Training:
  - We can train similar to language models:
  - Maximum likelihood approach

Downsides?

- Downsides:
  - Slow sequential generation process
  - Only considers few context pixels

Oord et al., Pixel Recurrent Neural Networks



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

- **Idea:** Represent conditional distribution as a convolution layer!
  - Because of spatial locality in images
- Considers larger context (receptive field)
- Practically can be implemented by applying a mask, zeroing out “future” pixels
- Faster training but still slow generation
  - Limited to smaller images

Oord et al., Conditional Image Generation with PixelCNN Decoders

Pixel CNN

Oord et al., Conditional Image Generation with PixelCNN Decoders

Example Results: Image Completion (PixelRNN)

Georgia Tech

Geyser

Hartebeest

Grey whale

Tiger

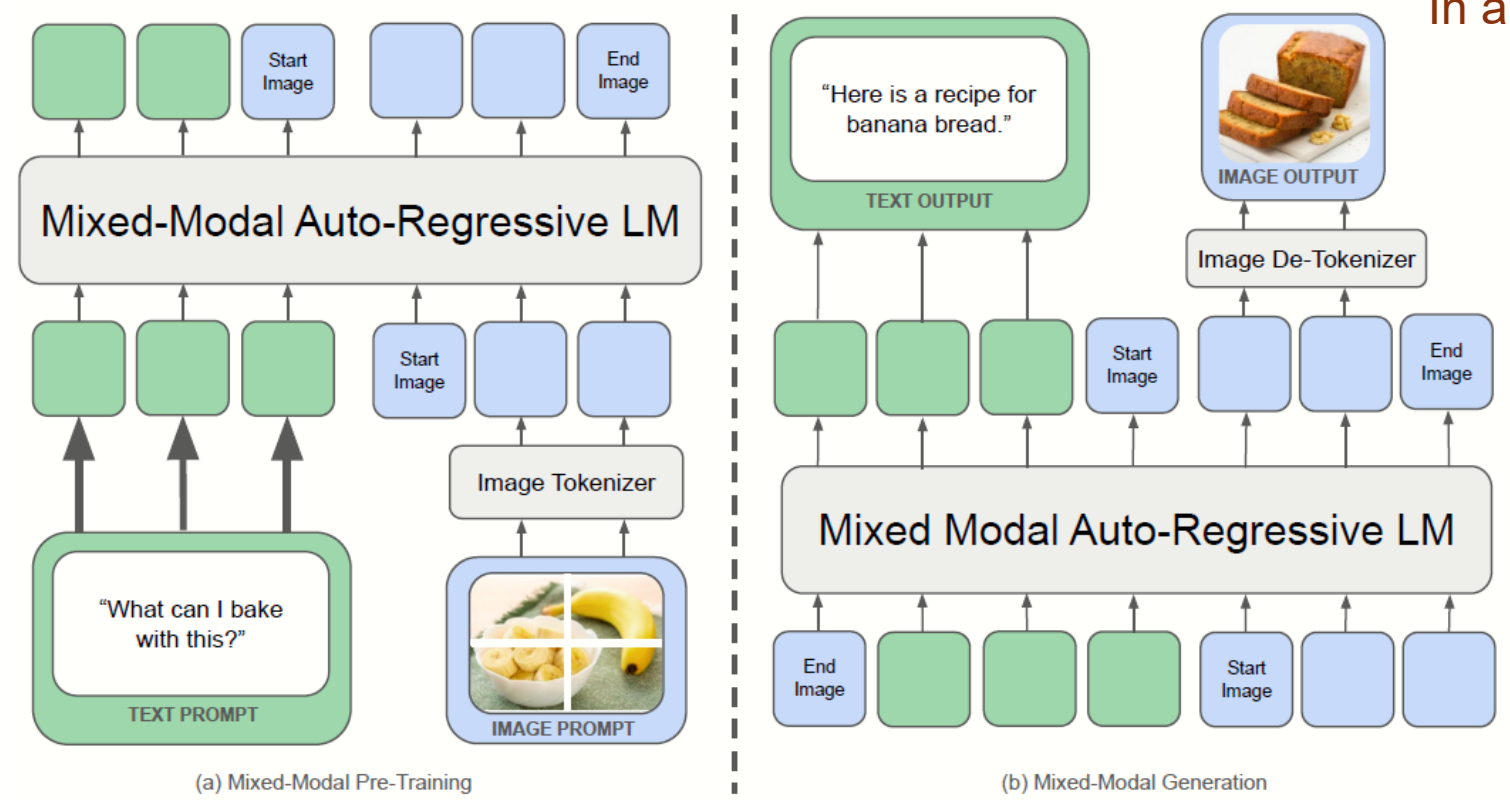
Can we update this to modern times?

Oord et al., Conditional Image Generation with PixelCNN Decoders

Example Images (PixelCNN)

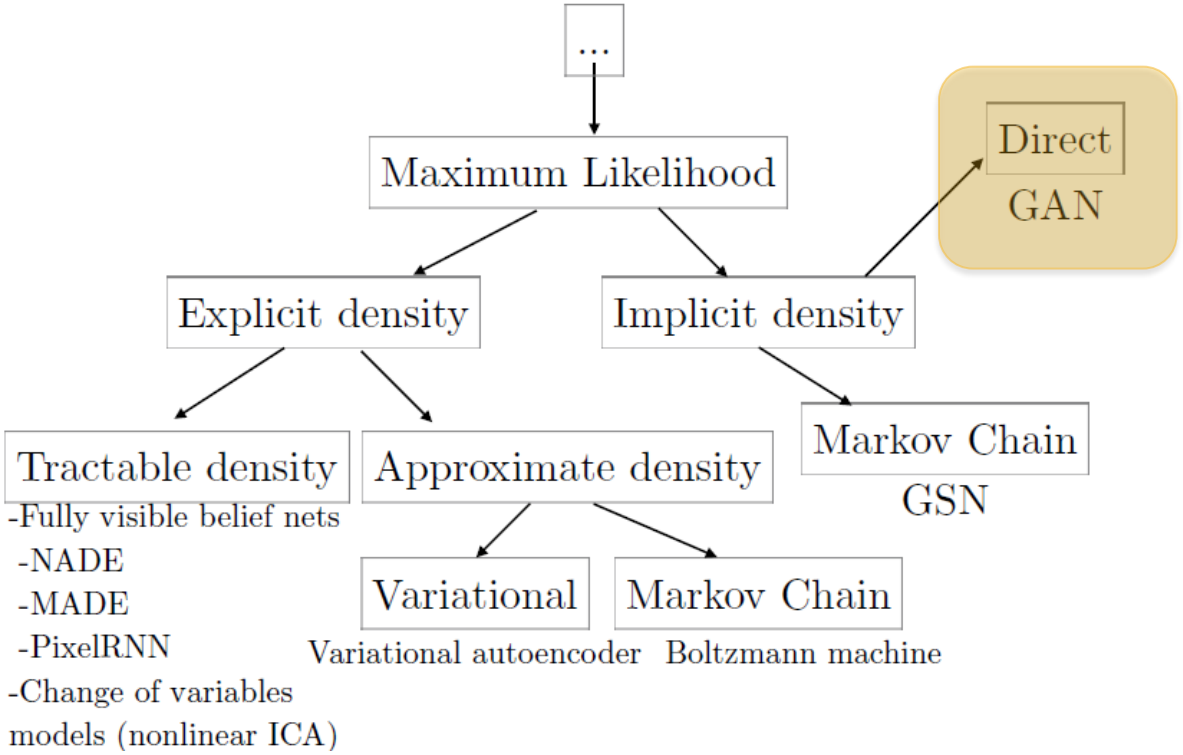
Georgia Tech

In a few weeks



Chameleon: Mixed-Modal Early-Fusion Foundation Models

Generative Adversarial Networks (GANs)



Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generative Models

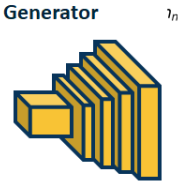
- *Implicit* generative models do not actually learn an explicit model for $\boldsymbol{p(x)}$

- Instead, learn to *generate samples* from $\boldsymbol{p(x)}$
  - Learn good feature representations
  - Perform data augmentation
  - Learn world models (a simulator!) for reinforcement learning

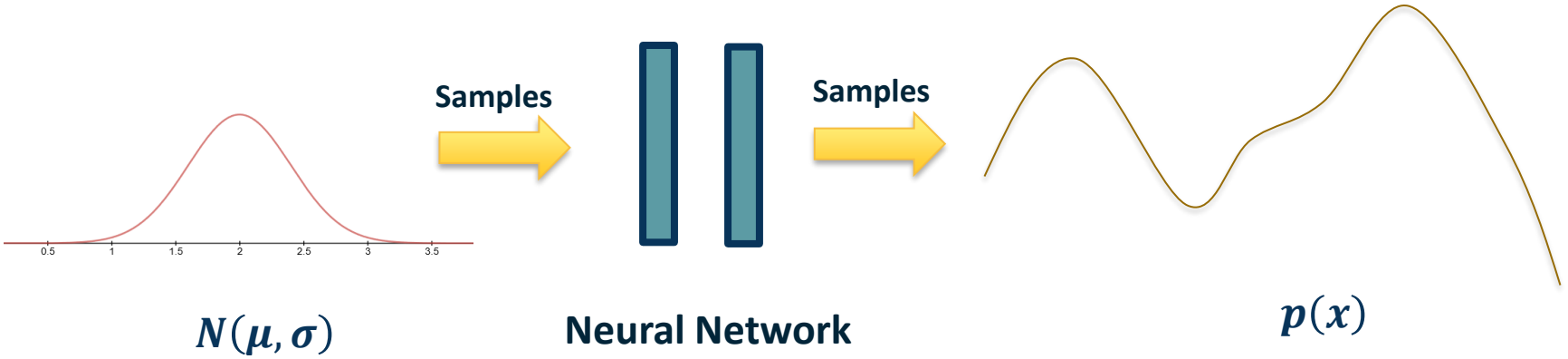
- How?
  - **Decode architecture**
  - **Learn to sample** from a neural network output

What architecture lets us generate images?

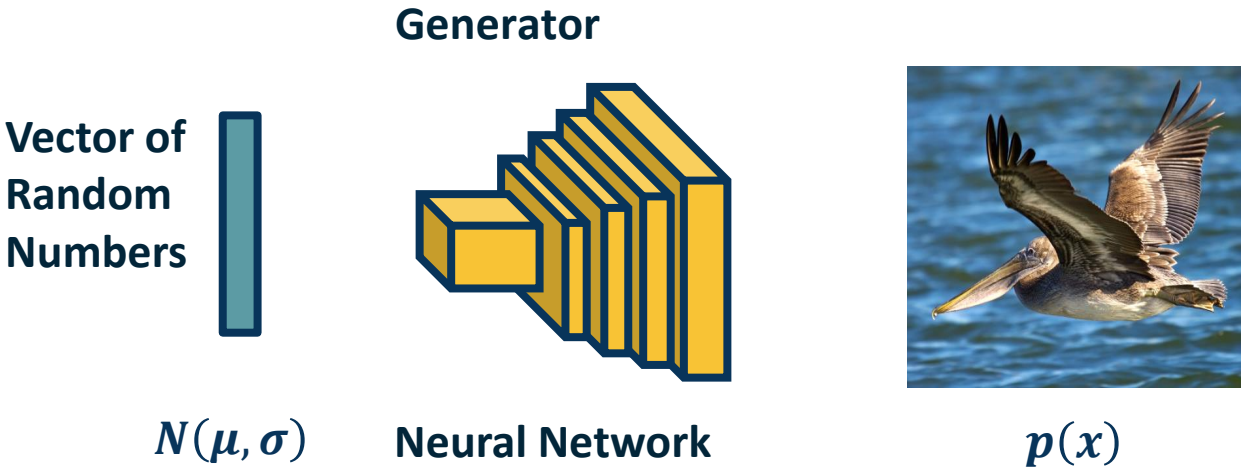
How do we generate a different image every time?



- We would like to *sample* from $\boldsymbol{p(x)}$ using a neural network
- **Idea:**
  - Sample from a simple distribution (Gaussian)
  - Transform the sample to $\boldsymbol{p(x)}$



- Input can be a vector with (independent) Gaussian random numbers
- We can use a CNN to generate images!

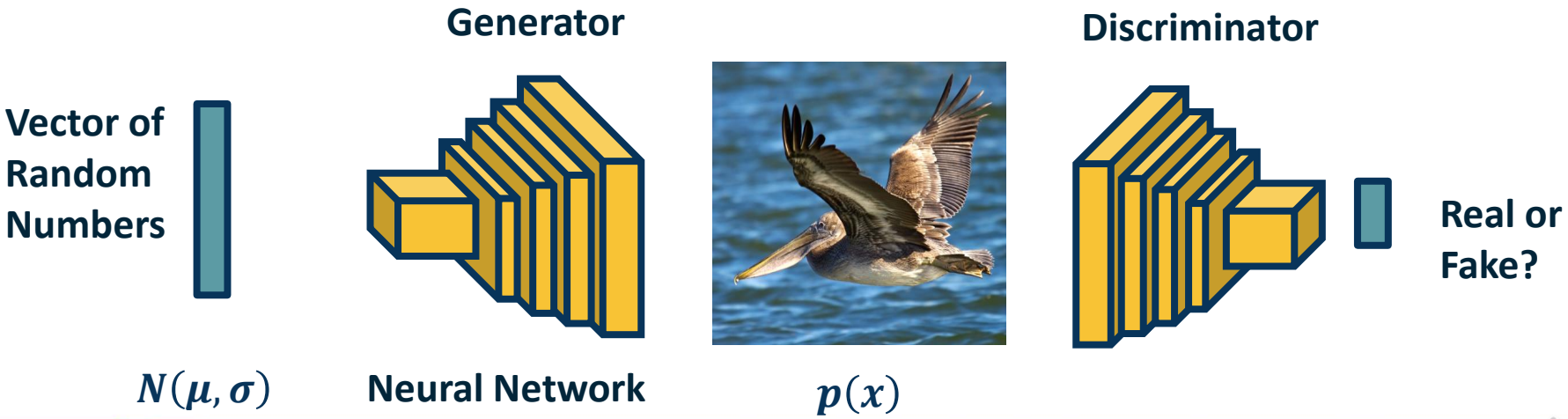


How do we train this (loss)?

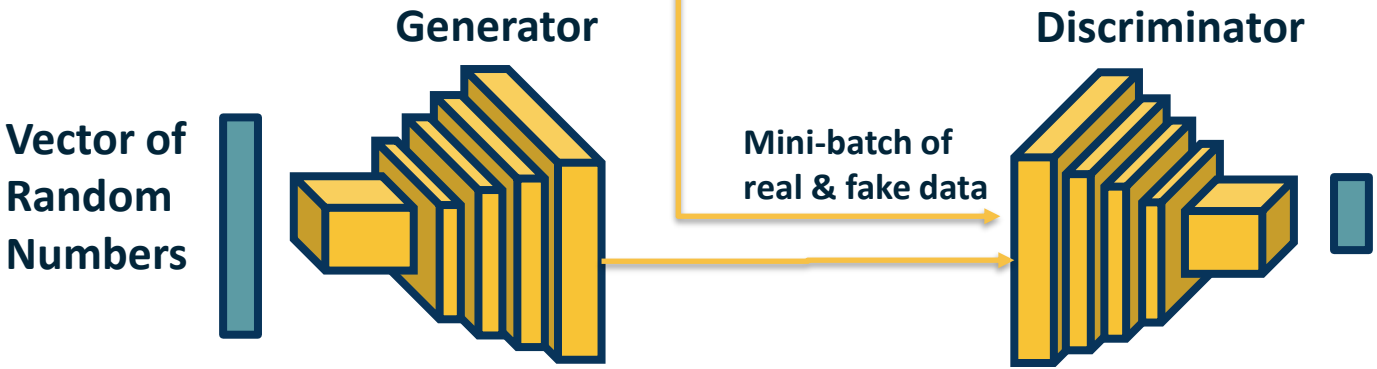
- Instead, learn to *generate samples* from $\boldsymbol{p(x)}$

- How?
  - **Adversarial training** that uses one network's predictions to train the other (dynamic loss function!)
  - **Lots of tricks** to make the optimization more stable

- **Goal:** We would like to generate *realistic* images. How can we drive the network to learn how to do this?
- **Idea:** Have *another* network try to distinguish a real image from a generated (fake) image
  - **Why?** Signal can be used to determine how well it's doing at generation
    - Can be seen as a dynamic (adversarial) loss!



- **Generator:** Update weights to improve realism of generated images
- **Discriminator:** Update weights to better discriminate



Question: What loss functions can we use (for each network)?

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Mini-max Two Player Game

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

- The full mini-max objective is:

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

Sample from real **Sample from fake**

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Mini-max Two Player Game

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

- where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
- x is a **real image** and $G(z)$ is a **generated** image

- The discriminator wants to **maximize** this:
  - $D(x)$ is pushed up (to 1) because x is a real image
  - $1 - D\big(G(z)\big)$ is also pushed up to 1 (so that $\mathrm{D}\big(\mathrm{G}(\mathrm{z})\big)$ is pushed down to 0)
  - In other words, discriminator wants to classify real images as real (1) and fake images as fake (0)

Discriminator Perspective

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

- where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
- x is a **real image** and $G(z)$ is a **generated** image

- The generator wants to **minimize** this:
  - First term: G(..) doesn't appear in it!
  - $1 - D\big(G(z)\big)$ is pushed down to 0 (so that $\mathrm{D}\big(\mathrm{G}(\mathrm{z})\big)$ is pushed up to 1)
  - This means that the generator is **fooling** the discriminator, i.e. succeeding at generating images that the discriminator can't discriminate from real

Generator Perspective

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

- The full mini-max objective is:

Sample from fake

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

Generator *minimizes*

How well discriminator does (0 for fake)

  - where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
  - x is a **real image** and $G(z)$ is a **generated** image

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

- Since we have two networks competing, this is a mini-max two player game
  - Ties to game theory
  - Not clear what (even local) Nash equilibria are for this game

- The full mini-max objective is:

Sample from real **Sample from fake**

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

Discriminator *maximizes*

How well discriminator does (1 for real)

How well discriminator does (0 for fake)

  - where $D(x)$ is the discriminator outputs probability ([0,1]) of **real** image
  - x is a **real image** and $G(z)$ is a **generated** image

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks

Generator

Discriminator

Vector of Random Numbers

Mini-batch of real & fake data

Cross-entropy (Real or Fake?)
We know the answer (self-supervised)

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

Generator Loss

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log \left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

Discriminator Loss

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, NeurIPS 2016 Generative Adversarial Nets

Final Algorithm

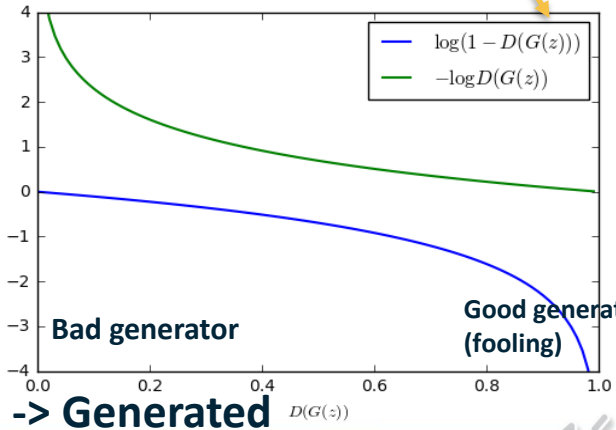
- The generator part of the objective does not have good gradient properties

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

  - High gradient when $D(G(z))$ is high (that is, discriminator is wrong)
  - We want it to improve when samples are *bad* (discriminator is right)

- Alternative objective, ***maximize***:

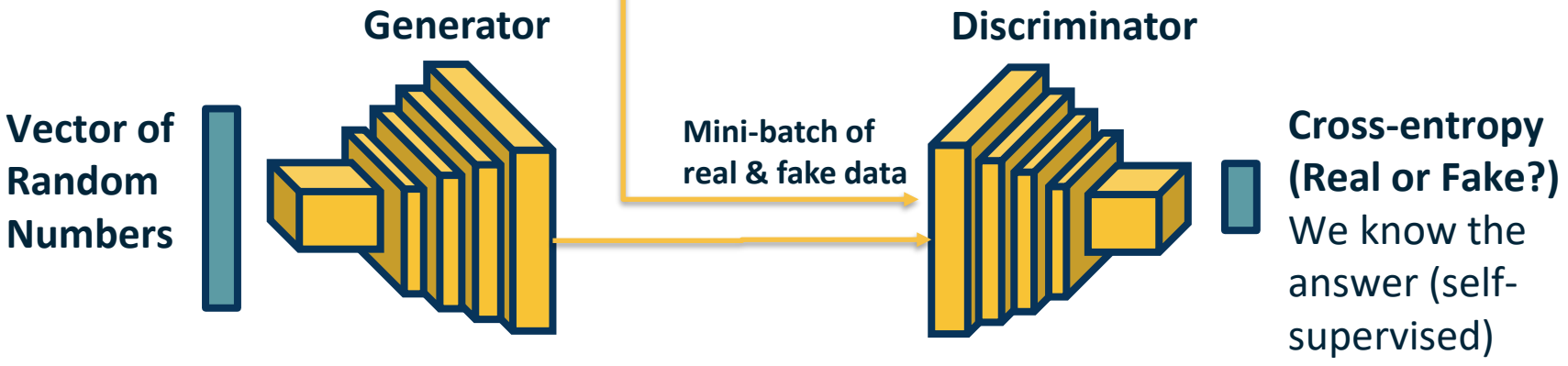
$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



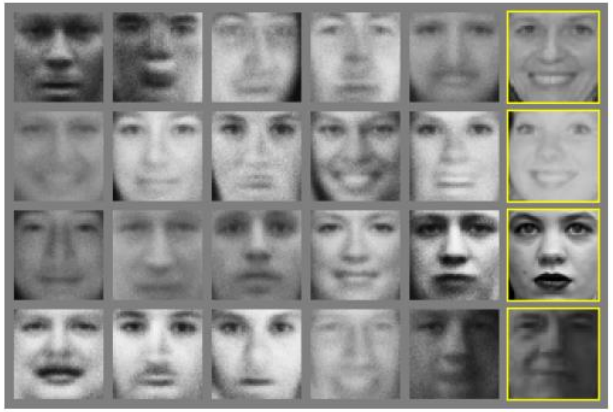
Plot from CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Converting to Max-Max Game

- At the end, we have:
  - An *implicit* generative model!
  - Features from discriminator

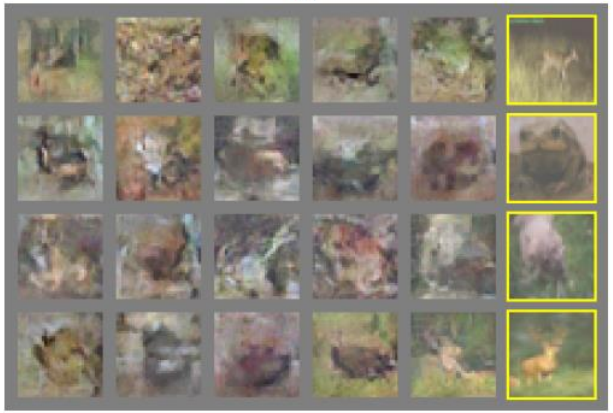


a)



b)

c)



d)

- Low-resolution images but look decent!
- Last column are nearest neighbor matches in dataset

Early Results

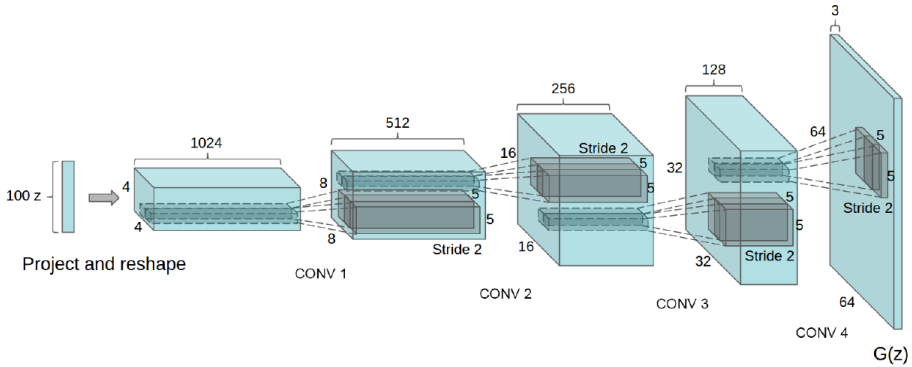
- GANs are very difficult to train due to the mini-max objective

- Advancements include:
  - More stable architectures
  - Regularization methods to improve optimization
  - Progressive growing/training and scaling

Goodfellow, NeurIPS 2016 Generative Adversarial Nets

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.



Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN

- Training GANs is difficult due to:
  - Minimax objective – For example, what if generator learns to memorize training data (no variety) or only generates part of the distribution?
  - Mode collapse – Capturing only some modes of distribution

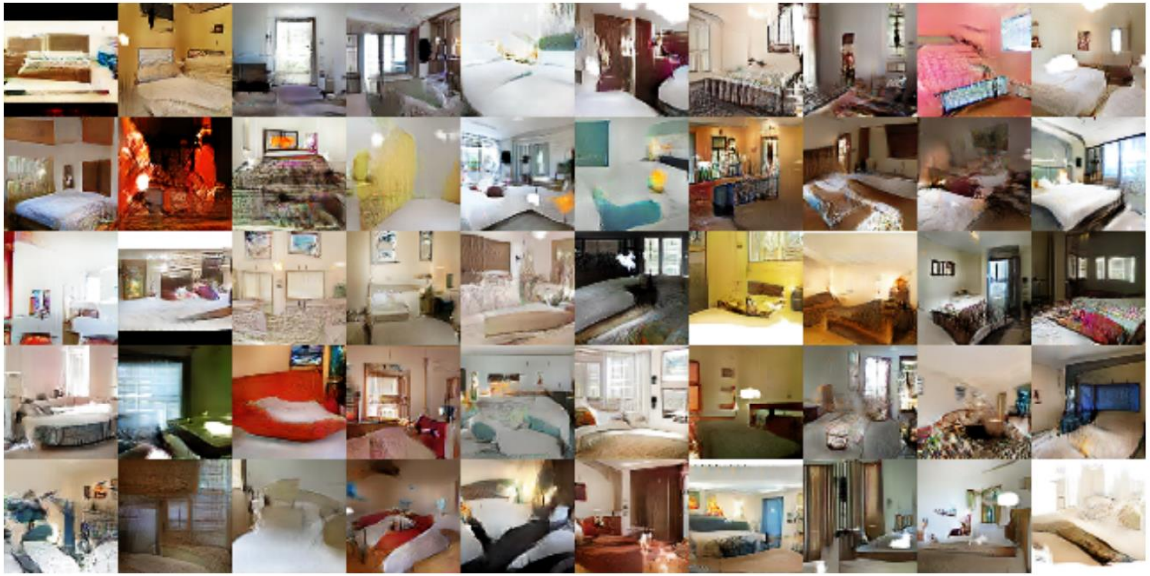
- Several theoretically-motivated regularization methods
  - Simple example: Add noise to real samples!

$$\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0, cI)} \left[\|\nabla_{\mathbf{x}} D_\theta(x + \delta)\| - k \right]^2$$

Kodali et al., On Convergence and Stability of GANs (also known as How to Train your DRAGAN)

Generative Adversarial Nets: Convolutional Architectures

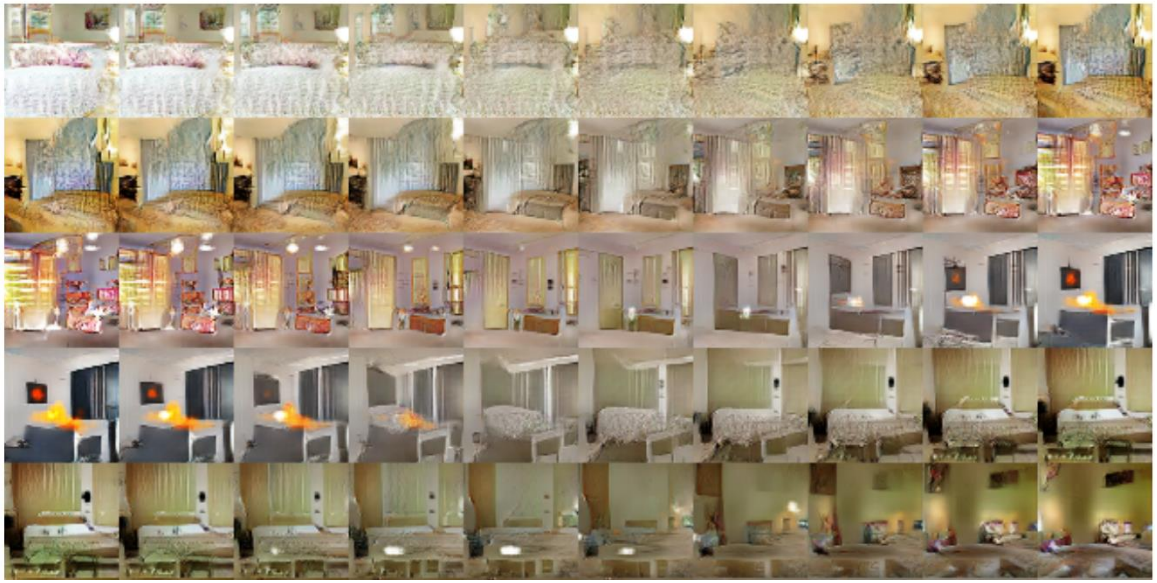
Samples from the model look much better!



Radford et al, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space



Radford et al, ICLR 2016

Brock et al., Large Scale GAN Training for High Fidelity Natural Image Synthesis

Example Generated Images - BigGAN

Georgia Tech

(a) 128×128 (b) 256×256 (c) 512×512 (d)

Figure 4: Samples from our model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).

Brock et al., Large Scale GAN Training for High Fidelity Natural Image Synthesis

Failure Examples - BigGAN

Georgia Tech

https://www.youtube.com/watch?v=PCBTZh41Ris

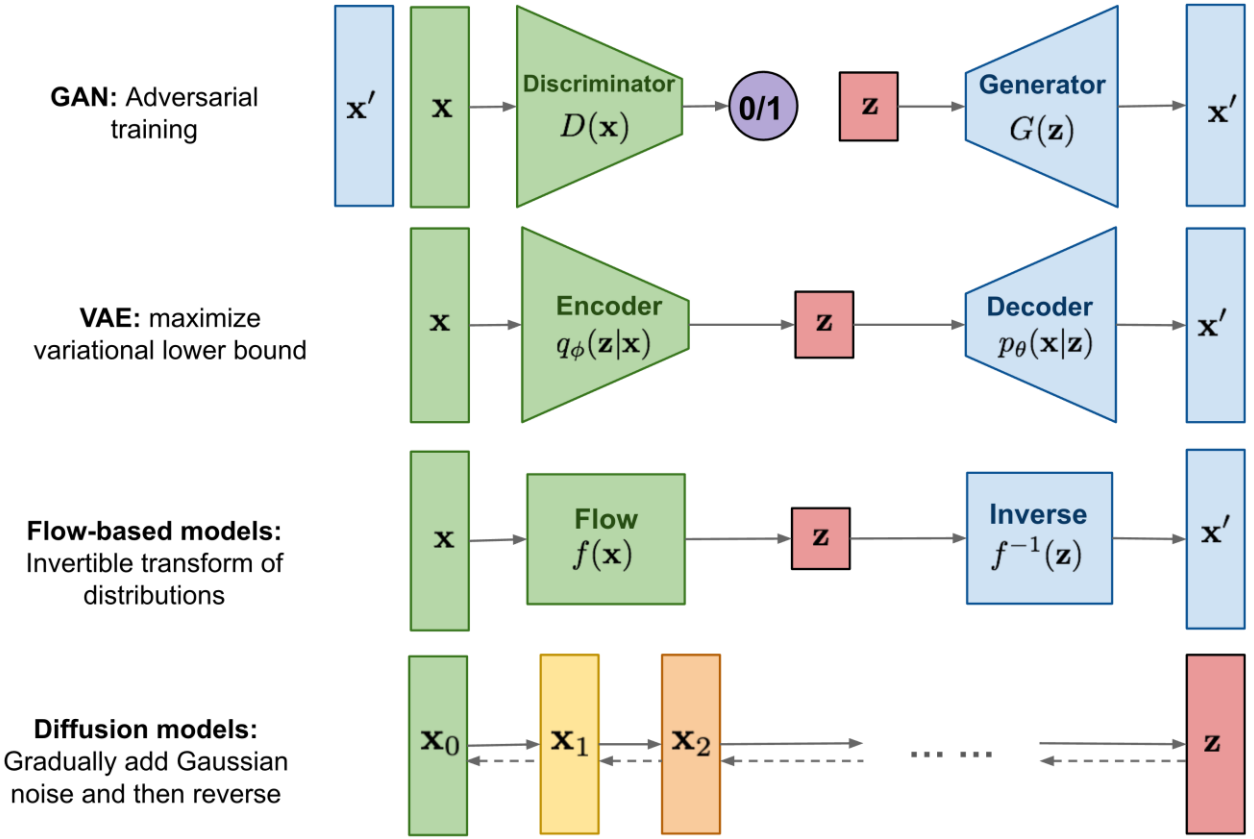
Video Generation

- Generative Adversarial Networks (GANs) can produce amazing images!

- Several drawbacks
  - High-fidelity generation heavy to train
  - Training can be unstable
  - No explicit model for distribution

- Larger number of extensions:
  - GANs conditioned on labels or other information
  - Adversarial losses for other applications

Comparison of Methods



LLaVA-OneVision: Easy Visual Task Transfer

Bo Li[2,♡] **Yuanhan Zhang**[2,♡] **Dong Guo**[1] **Renrui Zhang**[3,♡] **Feng Li**[4,♡] **Hao Zhang**[4,♡]
Kaichen Zhang[2] **Peiyuan Zhang**[2] **Yanwei Li**[3,♡] **Ziwei Liu**[2] **Chunyuan Li**[1]

[1]ByteDance [2]S-Lab, NTU [3]CUHK [4]HKUST

https://llava-vl.github.io/blog/llava-onevision

Abstract

We present LLaVA-OneVision, a family of open large multimodal models (LMMs) developed by consolidating our insights into data, models, and visual representations in the LLaVA-NeXT blog series. Our experimental results demonstrate that LLaVA-OneVision is the first single model that can simultaneously push the performance boundaries of open LMMs in three important computer vision scenarios: single-image, multi-image, and video scenarios. Importantly, the design of LLaVA-OneVision allows strong transfer learning across different modalities/scenarios, yielding new emerging capabilities. In particular, strong video understanding and cross-scenario capabilities are demonstrated through task transfer from images to videos.

Problem Statement

- What problem does this paper focus on?
  - Is this new or already explored?
  - Is this important?
  - What key applications this is relevant for?
  - What assumptions does this paper make about

Related Work / Situation of Work

- What prior approaches exist to solve this problem?

The SoTA proprietary LMMs, such as GPT-4V [109], GPT-4o [110], Gemini [131] and Claude-3.5 [3], exhibit excellent performance in versertile vision scenarios, including single-image, multi-image and video settings. In the open research community, existing works typically develop models tailored to each individual scenario separately. Specifically, most focus on pushing the performance limits in single-image scenarios [26, 83, 173, 73, 164, 35], only a few recent papers have begun to explore multi-image scenarios [70, 47]. While video LMMs excel in video understanding, they often do so at the expense of image performance [72, 76]. It is rare to have a single open model that reports excellent performance in all three scenarios. LLaVA-OneVision aims to fill this gap by demonstrating state-of-the-art performance across a broad range of tasks, and showcasing interesting emerging capabilities through cross-scenario task transfer and composition.

To the best of our knowledge, LLaVA-NeXT-Interleave [68] is the first attempt to report good performance in all three scenarios, LLaVA-OneVision inherits its training recipe and data for improved

Related Work / Situation of Work

- Built on LLaVA-NeXT

Compared with LLaVA-1.5, LLaVA-NeXT has several improvements:

1. **Increasing the input image resolution** to 4x more pixels. This allows it to grasp more visual details. It supports three aspect ratios, up to 672x672, 336x1344, 1344x336 resolution.
2. **Better visual reasoning and OCR capability** with an improved visual instruction tuning data mixture.
3. **Better visual conversation for more scenarios**, covering different applications. Better world knowledge and logical reasoning.
4. **Efficient deployment and inference** with SGLang

Related Work / Situation of Work

(1) Dynamic High Resolution

We design our model at high resolution with an aim to **preserve its data efficiency**. When provided with high-resolution images and representations that preserve these details, the model's capacity to perceive intricate details in an image is significantly improved. It reduces the model hallucination that conjectures the imagined visual content when confronted with low-resolution images. Our 'AnyRes' technique is designed to accommodate images of various high resolutions. We employ a grid configuration of $\{2 \times 2, 1 \times \{2, 3, 4\}, \{2, 3, 4\} \times 1\}$, balancing performance efficiency with operational costs. See our updated LLaVA-1.5 technical report for more details.

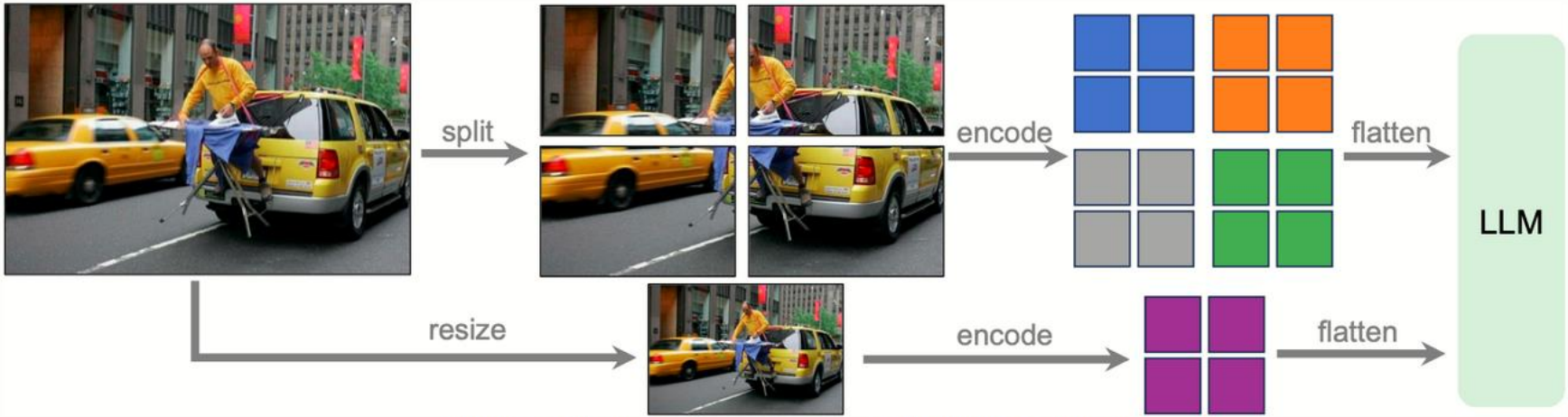


Illustration of dynamic high resolution scheme: a grid configuration of 2×2

Approach and Key Nugget

- What approach does this paper take?
- What is the key “golden nugget” – intuition, idea, etc. that leads to approach
  - Lower-level
  - Higher-level?

Contributions

recipe, Please see the detailed development timeline in Section A. In particular, our paper makes the following contributions:

- *Large multimodal models.* We develop LLaVA-OneVision, a family of open large multimodal models (LMMs) that improves the performance boundaries of open LMMs in three important vision settings, including single-image, multi-image, and video scenarios.
- *Emerging Capabilities with Task Transfer.* Our design in modeling and data representations allow task transfer across different scenarios, suggesting a simple approach to yield new emgerging capabilities. In particular, LLaVA-OneVision demonstrate strong video understanding through task transfer from images.
- *Open-source.* To pave the way towards building a general-purpose visual assistant, we release the following assets to the public: the generated multimodal instruction data, the codebase, the model checkpoints, and a visual chat demo.

Architecture

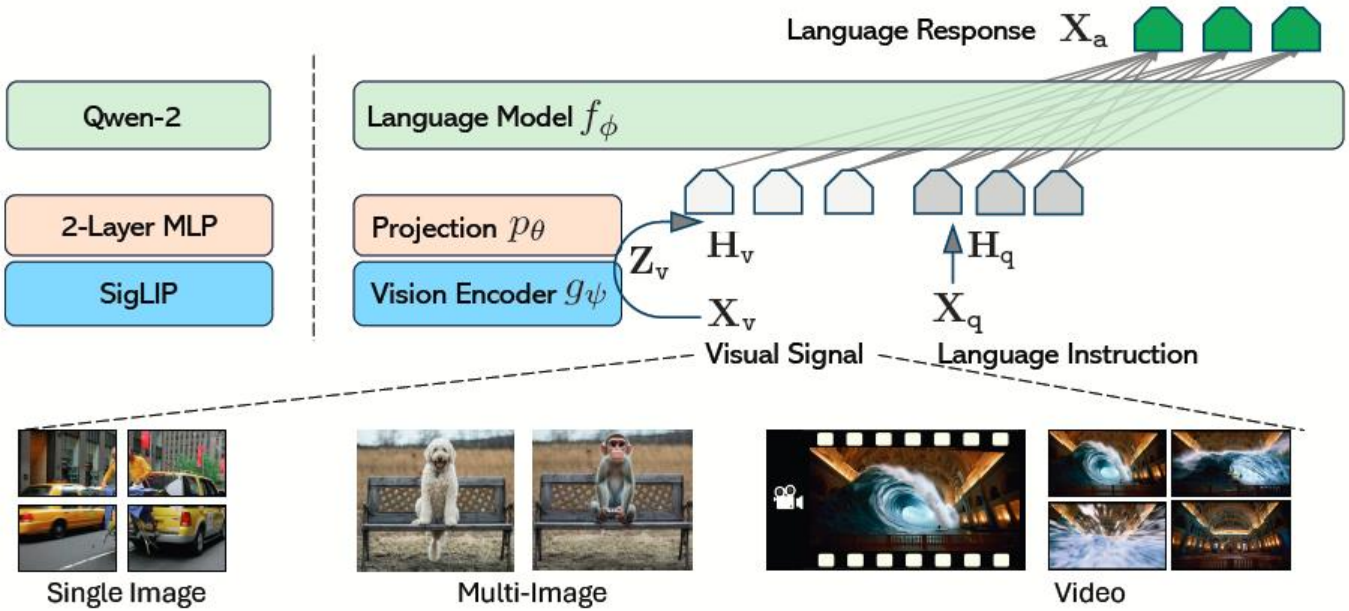
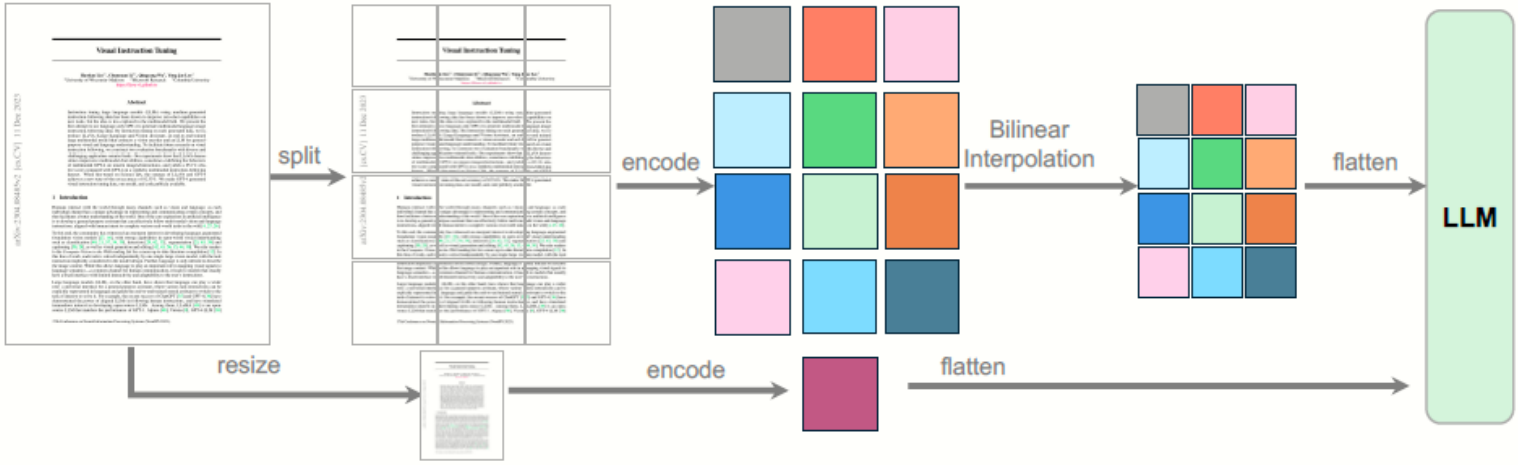


Figure 1: LLaVA-OneVision network architecture. Left: The current model instantiation; Right: the general form of LLaVA architecture in [83], but is extended to support more visual signals.

Multi-Resolution

To strike a balance of performance and cost, we observe that the scaling of resolution is more effective than that of token numbers, and recommend an AnyRes strategy with pooling.



(a) Higher AnyRes with Bilinear Interpolation

Data

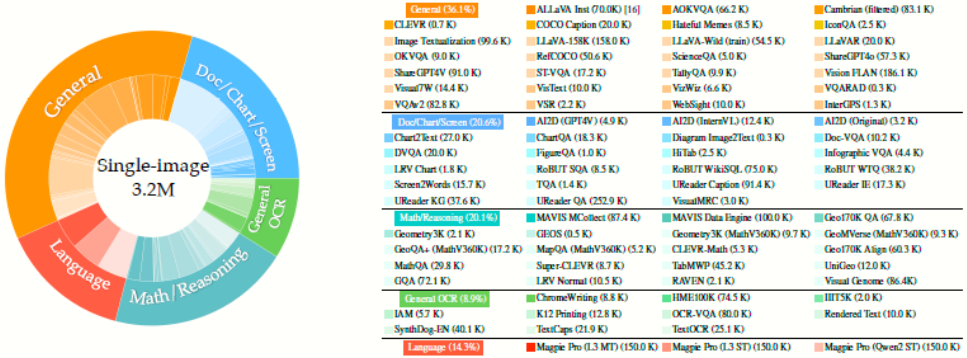


Figure 4: **Single-Image 3.2M.** A High-Quality Single-Image Dataset Collection. Left: Data Distribution within Each Category. The outer circle shows the distribution of all data categories and the inner circle shows the distribution of data subsets. Right: The detailed quantities of datasets.

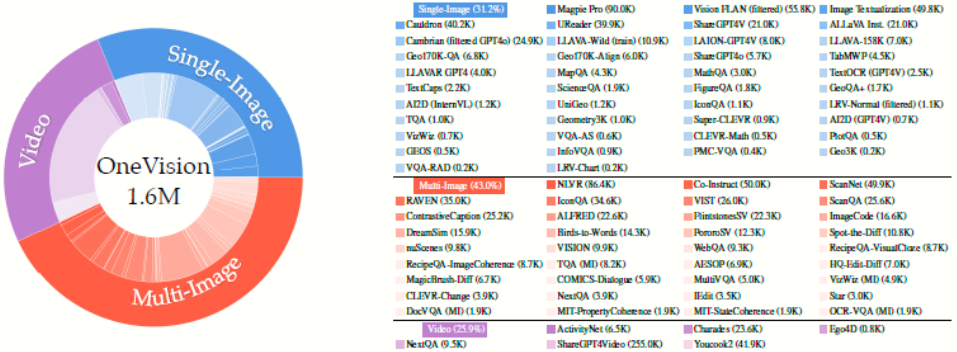
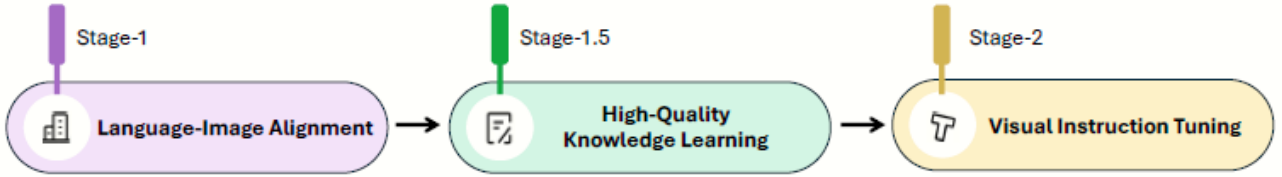


Figure 5: **OneVision 1.6M.** A high-quality single-image, multi-image and video dataset collection. Left: Data Distribution within each category. The outer circle shows the distribution of all data categories and the inner circle shows the distribution of data subsets. Right: The detailed quantities of datasets. "MI" means it is the multi-image version dataset proposed by DEMON [69].

Training

- *Stage-1: Language-Image Alignment.* The goal is to well align the visual features into the word embedding space of LLMs.
- *Stage-1.5: High-Quality Knowledge Learning.* To strike a balance between compute-efficiency and injecting new knowledge into LMMs, we recommend to consider the high-quality knowledge for LMM learning. The training configuration mirrors the settings used in Stage-2, ensuring consistency and allowing the model to integrate new information seamlessly.
- *Stage-2: Visual Instruction Tuning.* To teach LMM to solve a diverse set of visual task with preferred responces, we organize the instruction data into different groups, described in Section 4.2. The model is scheduled to train on these groups in order.

Training



		Stage-1	Stage-1.5	Stage-2	
				Single-Image	OneVision
Vision	**Resolution**	384	384×{2×2, 1×{2,3}, {2,3}×1}	384×{{1×1}, ···, {6×6}}	384×{{1×1}, ···, {6×6}}
	#Tokens	729	Max 729×5	Max 729×10	Max 729×10 (See Fig. 3)
Data	**Dataset**	LCS	Image (Sec. 4.1)	Image (Sec. 4.2)	(Multi)-Image & Video (Sec. 4.2)
	#Samples	558K	4M	3.2M	1.6M
Model	**Trainable**	Projector	Full Model	Full Model	Full Model
	0.5B LLM	1.8M	0.8B	0.8B	0.8B
	7.6B LLM	20.0M	8.0B	8.0B	8.0B
	72.7B LLM	72.0M	73.2B	73.2B	73.2B
Training	**Batch Size**	512	256/512	256/512	256/512
	LR: ψ_{vision}	1×10^{-3}	2×10^{-6}	2×10^{-6}	2×10^{-6}
	LR: $\{\theta_{\text{proj}}, \phi_{\text{LLM}}\}$	1×10^{-3}	1×10^{-5}	1×10^{-5}	1×10^{-5}
	Epoch	1	1	1	1

Table 1: Detailed configuration for each training stage of the LLaVA-OneVision model. The table outlines the progression of vision parameters, dataset characteristics, model specifications, and training hyperparameters across different stages of the curriculum learning process. We use a global batch size of 512 for the 0.5B model, and 256 for the 7B and 72B models.

Validation

- How do they validate their approach?
  - What data do they use?
  - What baselines do they compare against?

Capability	Benchmark	LLaVA OneVision-0.5B	LLaVA OneVision-7B	LLaVA OneVision-72B	GPT-4V (V-Preview)	GPT-4o
Single-Image	†AI2D [53] Science Diagrams	57.1%	81.4%	85.6%	78.2%	94.2%
	†ChartQA [101] Chart Understanding	61.4%	80.0%	83.7%	78.5%	85.7%
	†DocVQA [103] (test) Document Understanding	70.0%	87.5%	91.3%	88.4%	92.8%
	†InfoVQA [102] (test) Infographic Understanding	41.8%	68.8%	74.9%	-	-
	MathVerse [165] (vision-mini) Professional Math Reasoning	17.9%	26.2%	39.1%	32.8%	50.2%
	MathVista [90] (testmini) General Math Understanding	34.8%	63.2%	67.5%	49.9%	63.8%
	MMBench [86] (en-dev) Multi-discip	52.1%	80.8%	85.9%	75.0%	-
	MME [28] (cog./perp.) Multi-discip	240/1238	418/1580	579/1682	517/1409	-
	MMStar [19] Multi-discip	37.5%	61.7%	66.1%	57.1%	-
	MMMU [157] (val) College-level Multi-disp	31.4%	48.8%	56.8%	56.8%	69.1%
	MMVet [153] Multi-discip	29.1%	57.5%	63.7%	49.9%	76.2%
	SeedBench [66] (image) Multi-discip; Large-scale	65.5%	75.4%	78.0%	49.9%	76.2%
	†ScienceQA [93] High-school Science	67.2%	96.0%	90.3%	75.7%	-
	ImageDC [65] Image Detail Description	83.3%	88.2%	91.2%	91.5%	-
	RealworldQA [141] Realworld QA	55.6%	66.3%	71.9%	61.4%	-
	Vibe-Eval [112] Chanllenging Cases	33.8%	51.7%	50.7%	57.9%	63.1%
	MM-LiveBench [161] (2406) Internet Content Understanding	49.9%	77.1%	81.5%	-	92.4%
	LLaVA-Wilder [65] (small) Realworld Chat	55.0%	67.8%	72.0%	81.0%	85.9%
Multi-Image	LLaVA-Interleave [68] Out-domain	33.3%	64.2%	79.9%	60.3%	-
	MuirBench [135] Comprehenave Multi-image	25.5%	41.8%	54.8%	62.3%	-
	Mantis [47] Multi-image in the Wild	39.6%	64.2%	77.6%	62.7%	-
	BLINK [31] Unustal Visual Scenarios	52.1%	48.2%	55.4%	51.1%	-
	†Text-rich VQA [84] OCR, Webpage, Ducument	65.0%	80.1%	83.7%	54.5%	-
Video	ActivityNetQA [155] Spatio-Temporal Reasoning	50.5%	56.6%	62.3%	57.0%	-
	EgoSchema [98] Egocentric Video Understanding	26.8%	60.1%	62.0%	-	-
	PerceptionTest [115] Perception and Reasoning	49.2%	57.1%	66.9%	-	-
	SeedBench [66] (video) Multi-discip; Video	44.2%	56.9%	62.1%	60.5%	-
	LongVideoBench [138] (val) Long Video	45.8%	56.3%	63.2%	60.7%	66.7%
	MLVU [170] Long Video Understanding	50.3%	64.7%	68.0%	49.2%	64.6%
	MVBench [71] Multi-discip	45.5%	56.7%	59.4%	43.5%	-
	VideoChatGPT [97] Video Conversation	3.12	3.49	3.62	4.06	-
	VideoMME [29] Multi-discip	44.0%	58.2%	66.2%	59.9%	71.9%

Table 2: Performance comparison to state-of-the-art commercial models with our LLaVA-OneVision models (0.5B to 72B parameters) across diverse evaluation benchmarks spanning multiple modalities. † indicates that the training set has been observed in our data mixture.

Model	AI2D	ChartQA	DocVQA	InfoVQA	MathVerse	MathVista	MMBench	MME	MMMU
	test	test	val/test	val/test	mini-vision	testmini	en-dev	test	val
Qwen-VL-Max [8]	79.3	79.8	-/93.1	-	23.0	51.0	77.6	2281	51.4
Gemini-1.5-Pro [130]	94.4	87.2	-/93.1	-/81.0	-	63.9	-	-	62.2
Claude 3.5 Sonnet [3]	94.7	90.8	-/95.2	49.7	-	67.7	-	-	68.3
GPT-4V [109]	78.2	78.5*	-/88.4	-	32.8	49.9	75.0	517/1409	56.8
GPT-4o [110]	94.2	85.7	-/92.8	-	50.2	63.8	-	-	69.1
Cambrian-34B [133]	79.7	73.8	-/75.5	-	-	53.2	81.4	-	49.7
VILA-34B [77]	-	-	-	-	-	-	82.4	1762	51.9
IXC-2.5-7B [162]	81.5	82.2	-/90.9	-/70.0	20.0	59.6	82.2	2229	42.9
InternVL-2-8B [22]	83.8	83.3	-/91.6	-/74.8	27.5	58.3	81.7	2210	49.3
InternVL-2-26B [22]	84.5	84.9	-/92.9	-/75.9	31.3	59.4	83.4	2260	48.3
LLaVA-OV-0.5B (SI)	*54.2*	*61.0*	*75.0/71.2*	*44.8/41.3*	*17.3*	*34.6*	*43.8*	*272/1217*	*31.2*
LLaVA-OV-0.5B	57.1	61.4	73.7/70.0	46.3/41.8	17.9	34.8	52.1	240/1238	31.4
LLaVA-OV-7B (SI)	*81.6*	*78.8*	*89.3/86.9*	*69.9/65.3*	*26.9*	*56.1*	*81.7*	*483/1626*	*47.3*
LLaVA-OV-7B	81.4	80.0	90.2/87.5	70.7/68.8	26.2	63.2	80.8	418/1580	48.8
LLaVA-OV-72B (SI)	*85.1*	*84.9*	*93.5/91.8*	*77.7/74.6*	*37.7*	*66.5*	*86.6*	*563/1706*	*57.4*
LLaVA-OV-72B	85.6	83.7	93.1/91.3	79.2/74.9	39.1	67.5	85.9	579/1682	56.8

Model	MMVet	MMStar	S-Bench	S-QA	ImageDC	MMLBench	RealWorldQA	Vibe-Eval	LLaVA-W	L-Wilder
	test	test	image	test	test	2024-06	test	test	test	small
Qwen-VL-Max [8]	-	-	-	-	-	-	-	-	-	-
Gemini-1.5-Pro [130]	-	-	-	-	-	85.9	70.4	60.4	-	-
Claude 3.5 Sonnet [3]	75.4	-	-	-	-	92.3	59.9	66.2	102.9	83.1
GPT-4V [109]	49.9	57.1	49.9	75.7	91.5	-	61.4	57.9	98.0	81.0
GPT-4o [110]	76.2	-	76.2	-	92.5	92.4	58.6	63.1	106.1	85.9
Cambrian-34B [133]	-	-	-	85.6	-	-	67.8	-	-	-
VILA-34B [77]	53.0	-	75.8	-	-	-	-	81.3	-	-
IXC-2.5-7B [162]	51.7	59.9	75.4	-	87.5	-	67.8	45.2	78.1	61.4
InternVL-2-8B [22]	60.0	59.4	76.0	97.0	87.1	73.4	64.4	46.7	84.5	62.5
InternVL-2-26B [22]	65.4	60.4	76.8	97.5	91.0	77.2	66.8	51.5	99.6	70.2
LLaVA-OV-0.5B (SI)	*26.9*	*36.3*	*63.4*	*67.8*	*83.0*	*43.2*	*53.7*	*34.9*	*71.2*	*51.5*
LLaVA-OV-0.5B	29.1	37.5	65.5	67.2	83.3	49.9	55.6	33.8	74.2	55.0
LLaVA-OV-7B (SI)	*58.8*	*60.9*	*74.8*	*96.6*	*85.7*	*75.8*	*65.5*	*47.2*	*86.9*	*69.1*
LLaVA-OV-7B	57.5	61.7	75.4	96.0	88.9	77.1	66.3	51.7	90.7	67.8
LLaVA-OV-72B (SI)	*60.0*	*65.2*	*77.6*	*91.3*	*91.5*	*84.4*	*73.8*	*46.7*	*93.7*	*72.9*
LLaVA-OV-72B	63.7	66.1	78.0	90.3	91.2	81.5	71.9	50.7	93.5	72.0

Table 3: LLaVA-OneVision performance on single-image benchmarks. *GPT-4V reports 4-shot results on ChartQA. All results are reported as 0-shot accuracy.

Georgia Tech

Strengths / Weaknesses

- Strengths?

- Weakness / Limitations?