Edge AI Workshop

Lecture I - Computer Vision on the Edge

Overview

- Give a high-level overview of the Computer Vision field
- What are the current paradigms in the field?

- What are **image embeddings** and how are the useful?
- What is the role of Rust in the current Computer Vision landscape?

The Bigger Picture

Software 2.0

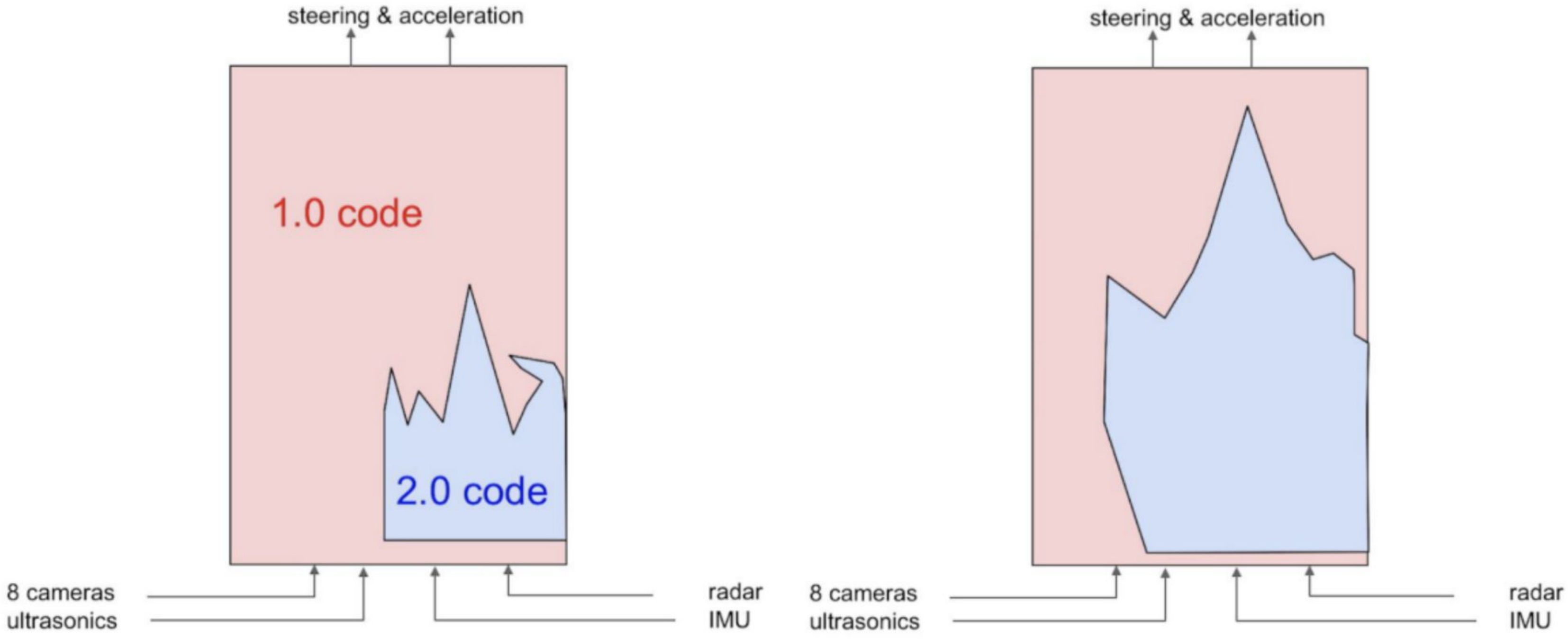
Software 1.0 = human-engineered source code (e.g. some .cpp files) is compiled into a binary that does useful work.

Software 2.0

source code is

- the dataset that defines the desirable behavior
- the neural net architecture that gives the rough skeleton of the code,
- many details (the weights) to be filled in

Software 2.0 starts to eat away at 1.0 codebases



Source: Andrej Karpathy

Analogy with Software 1.0

Source Code

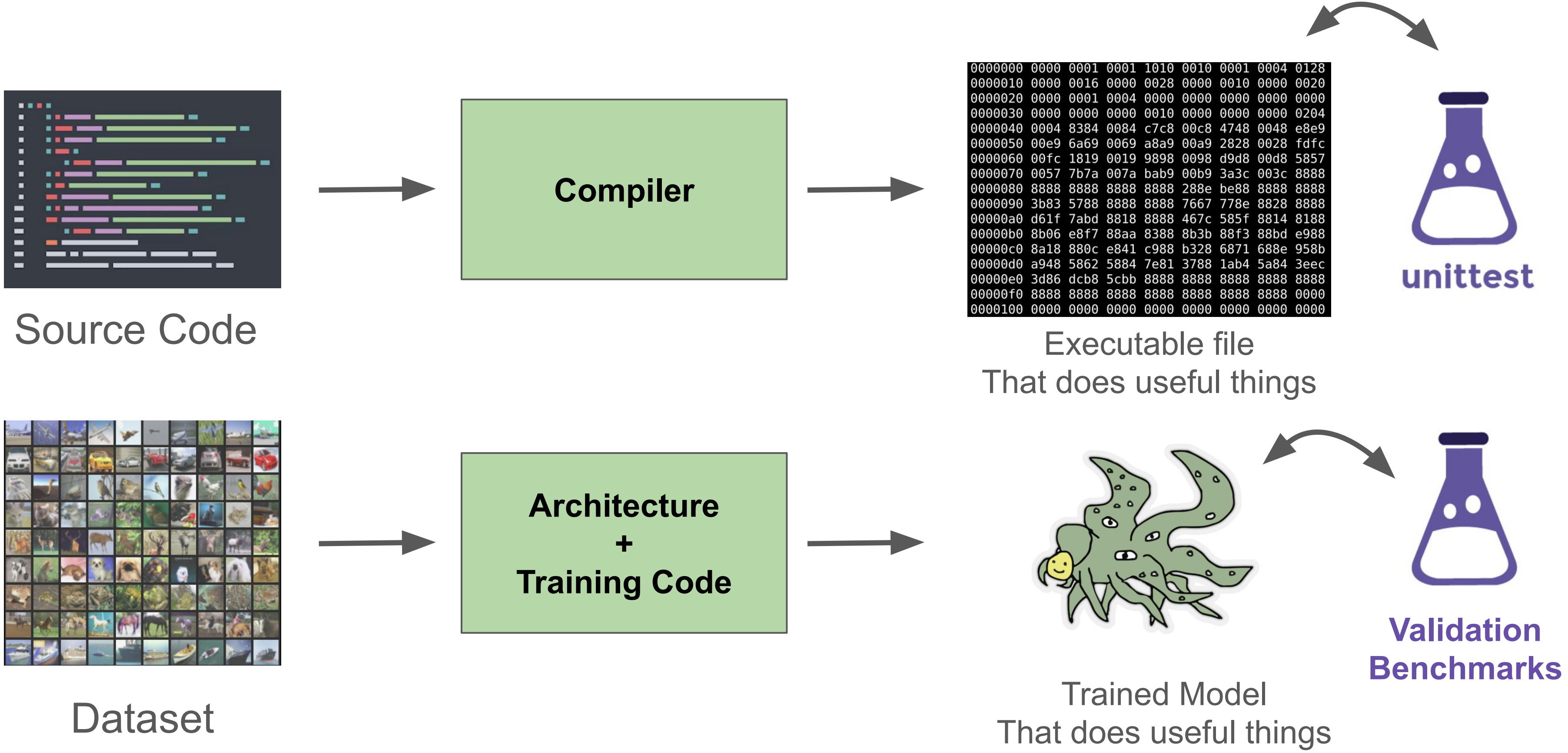
Compiler

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
```

Executable file
That does useful things

unittest

Analogy with Software 1.0



Andrej Karpathy
@karpathy

100% Fully Software 2.0 computer. Just a single neural net and no classical software at all. Device inputs (audio video, touch etc) directly feed into a neural net, the outputs of it directly display as audio/video on speaker/screen, that's it.

10:32 PM · Jun 30, 2024 · **807.9K** Views

Why is Edge AI appealing?

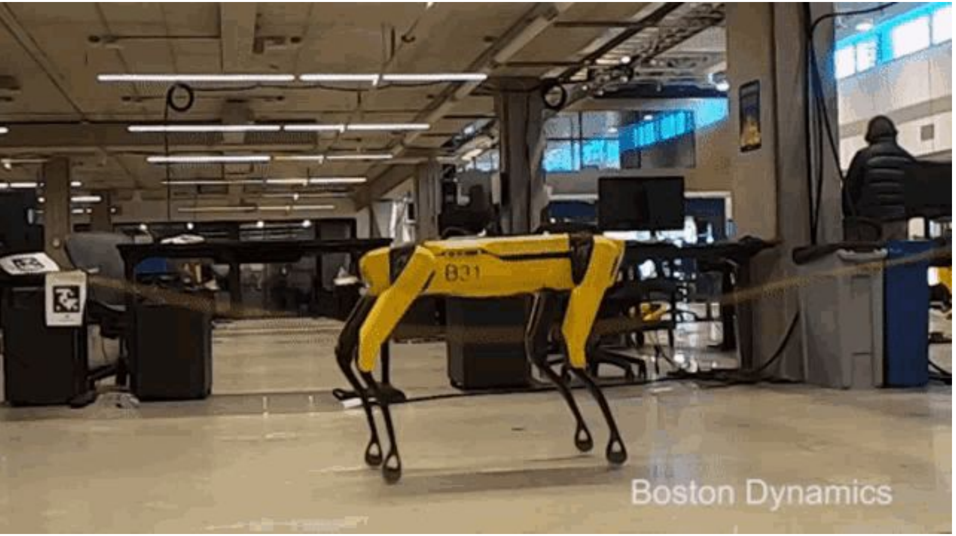
- Privacy preserving
  - no data leaves the device (hopefully)
- Decentralized
  - each device is independent / no need for a central server
- Scalable to billions of devices
- Enables more sophisticated methods of training
  - Federated learning - each device learns on its own data and then shares its knowledge

Why is EdgeAI hard?

- Small scale means smaller capability for models
- Latency & compute (real-time inference is hard)
- Low power consumption

- Edge devices are resource-constrained and need safe and efficient code
- **Rust is uniquely positioned to handle these requirements**

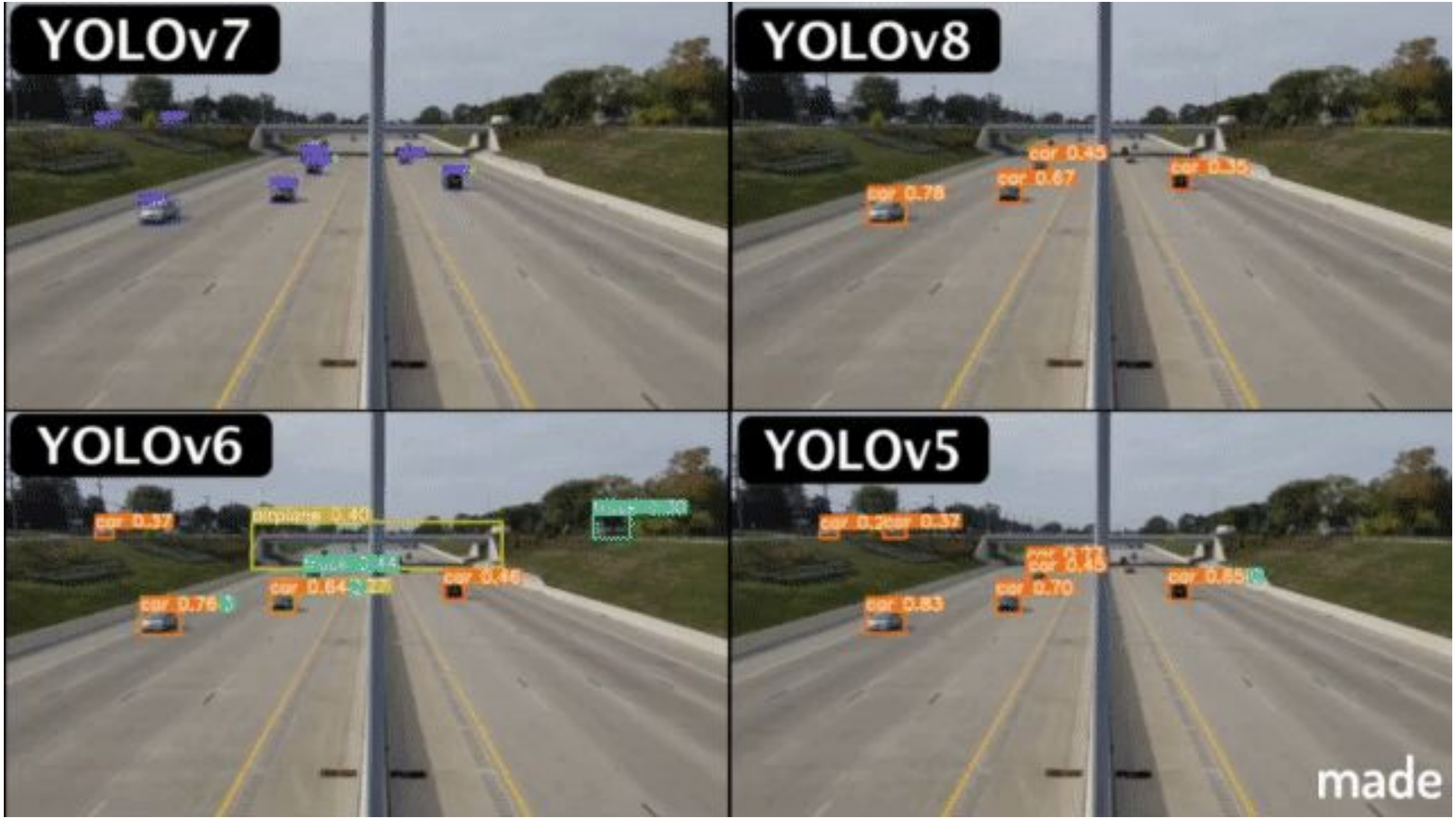
Computer Vision on the Edge - Robotics



Computer Vision on the Edge - Mobile

Computer Vision on the Edge - Autonomous Driving

Computer Vision on the Edge - Object Detection



Computer Vision on the Edge - Drones

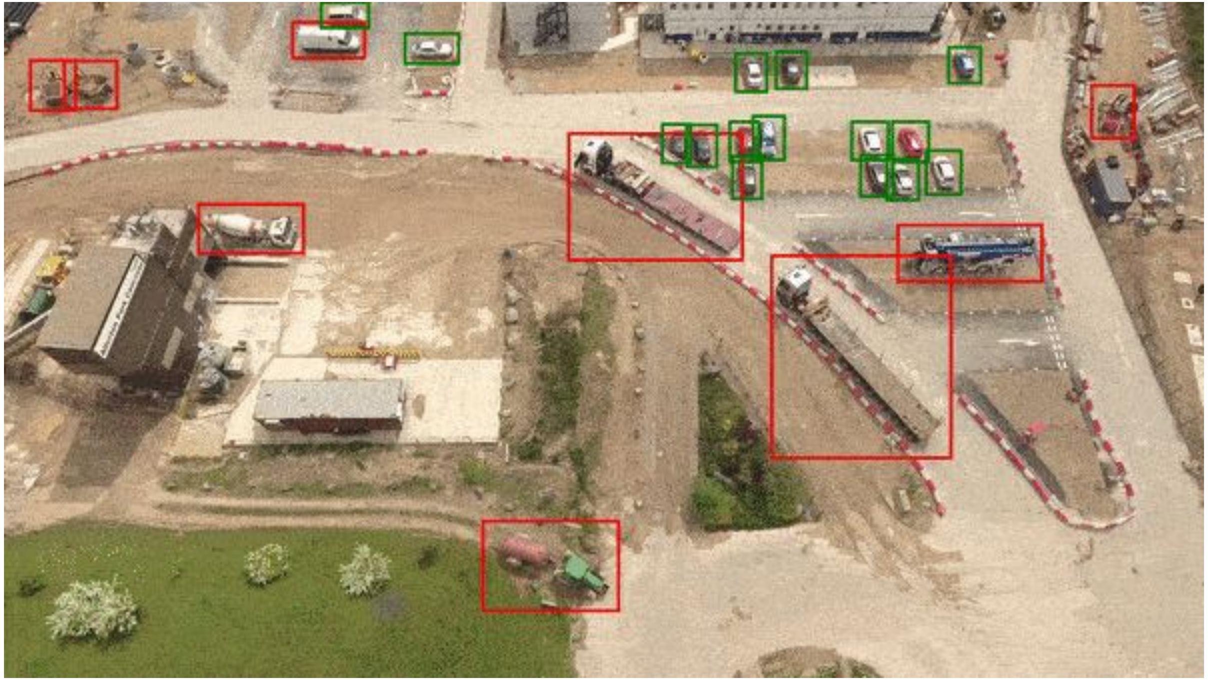
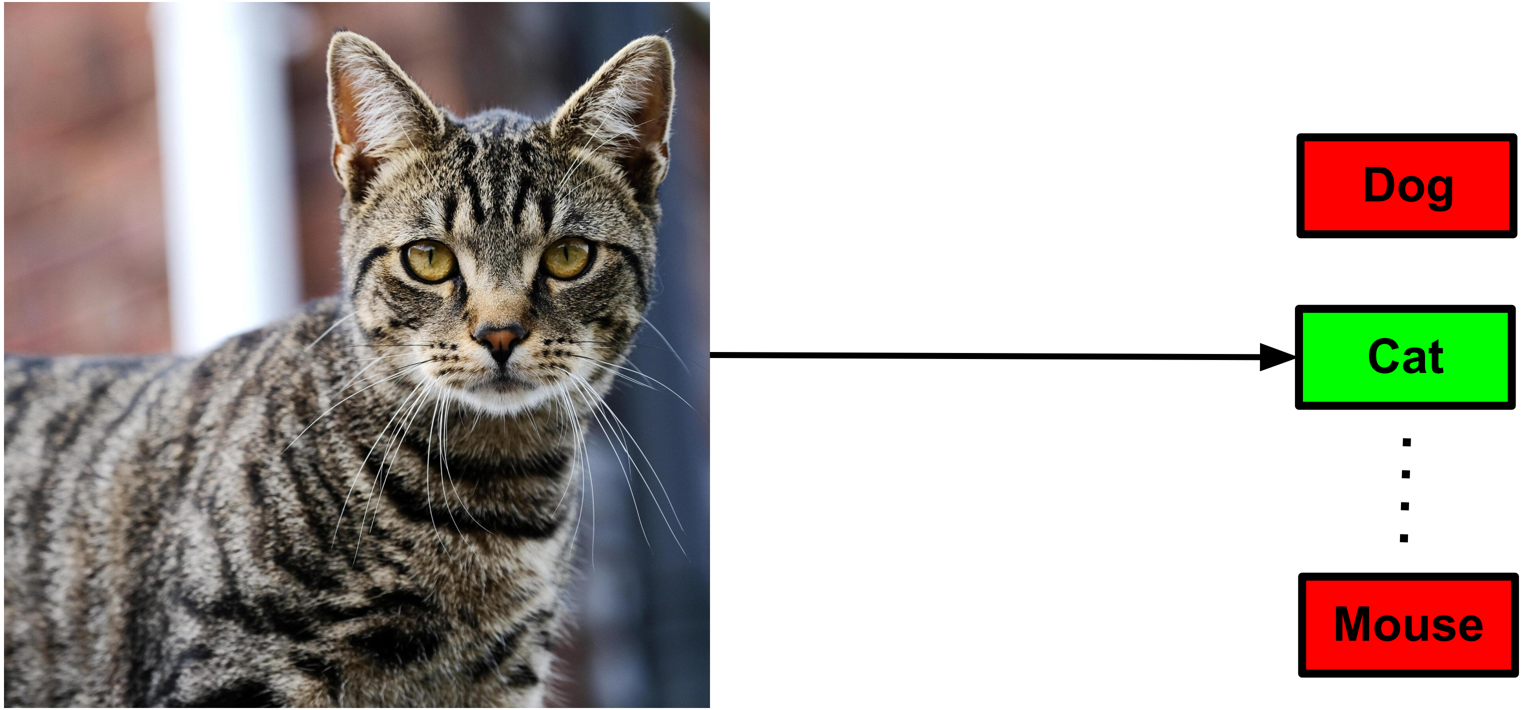
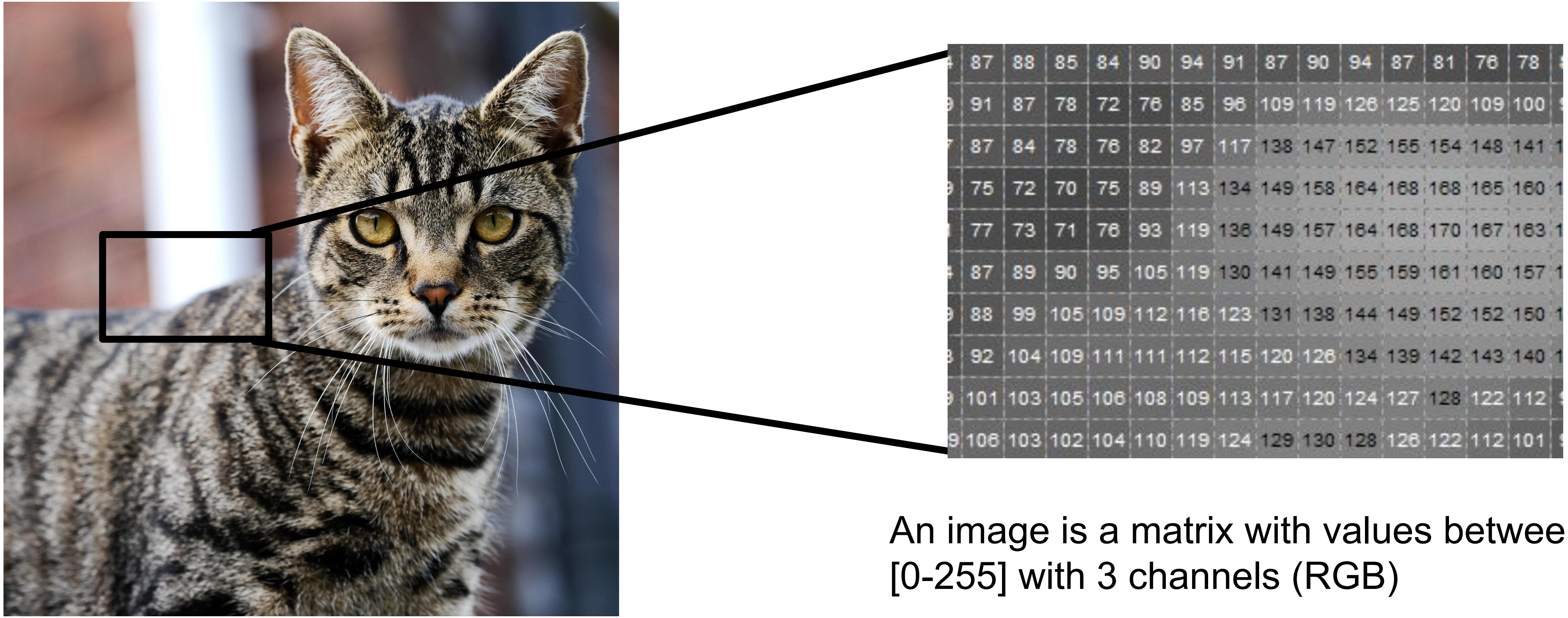


Image Classification - Core Task in CV



What does a Computer see?



An image is a matrix with values between [0-255] with 3 channels (RGB)

Minor changes to the image - all pixels differ

≠

Cats vary in a lot of ways

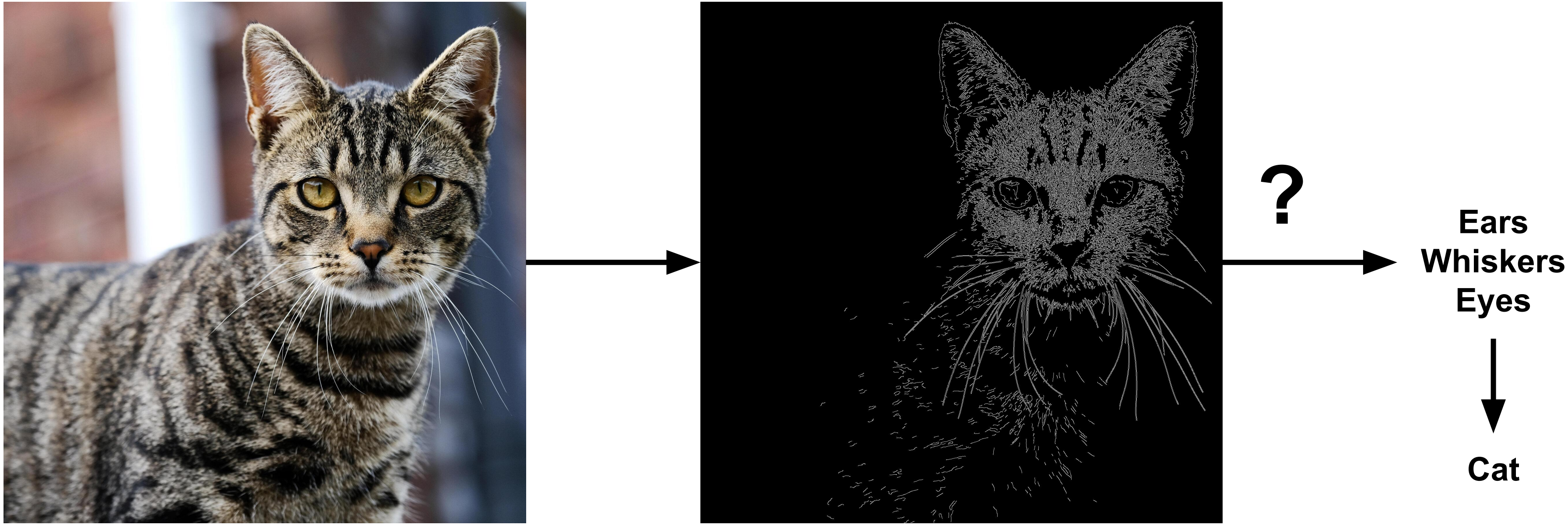
Other challenges

Occlusion

Illumination

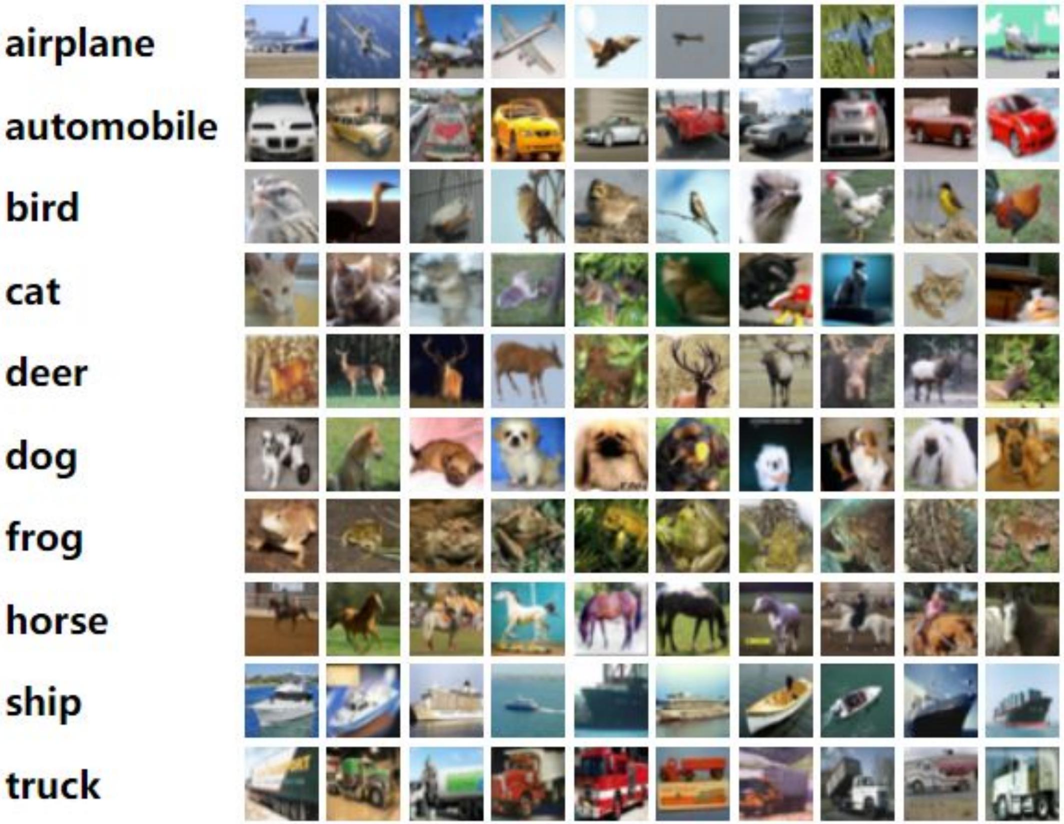
Deformation

Naive approach

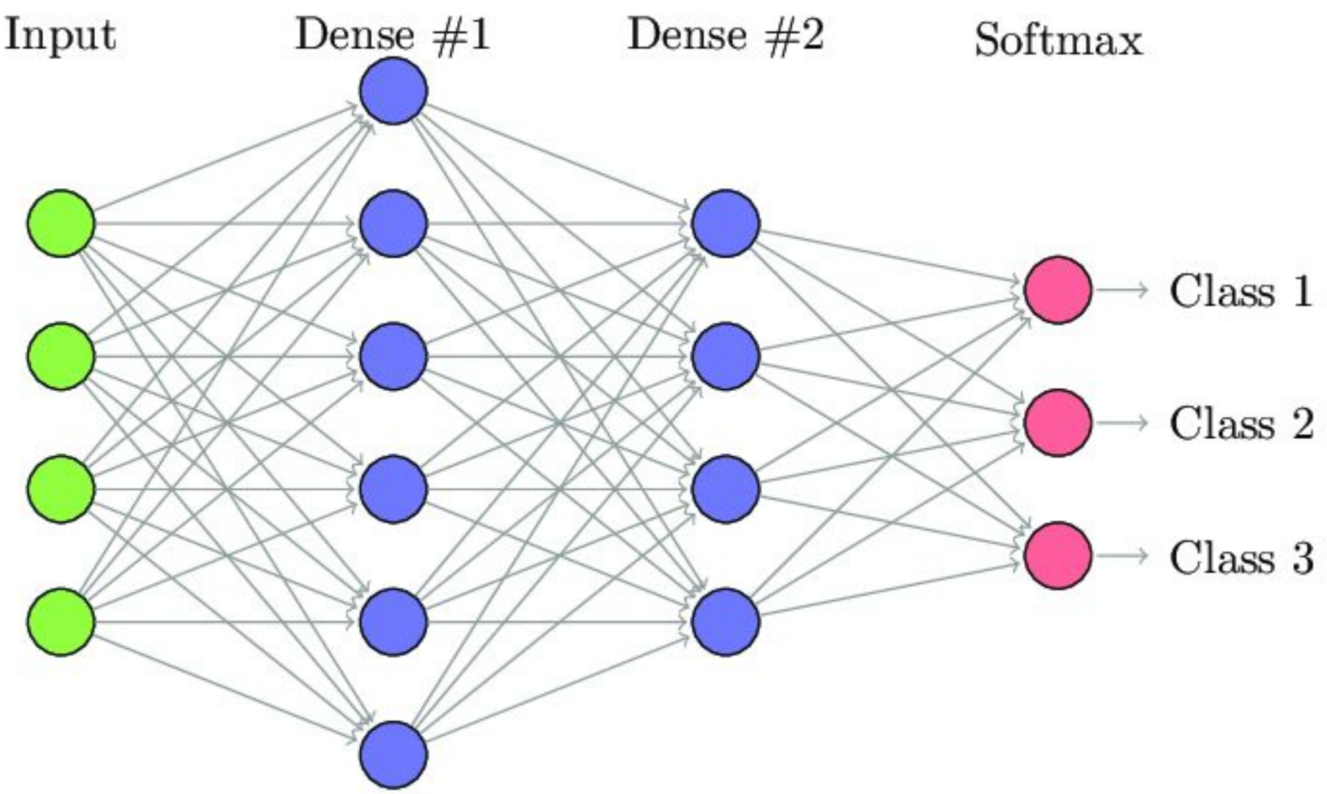


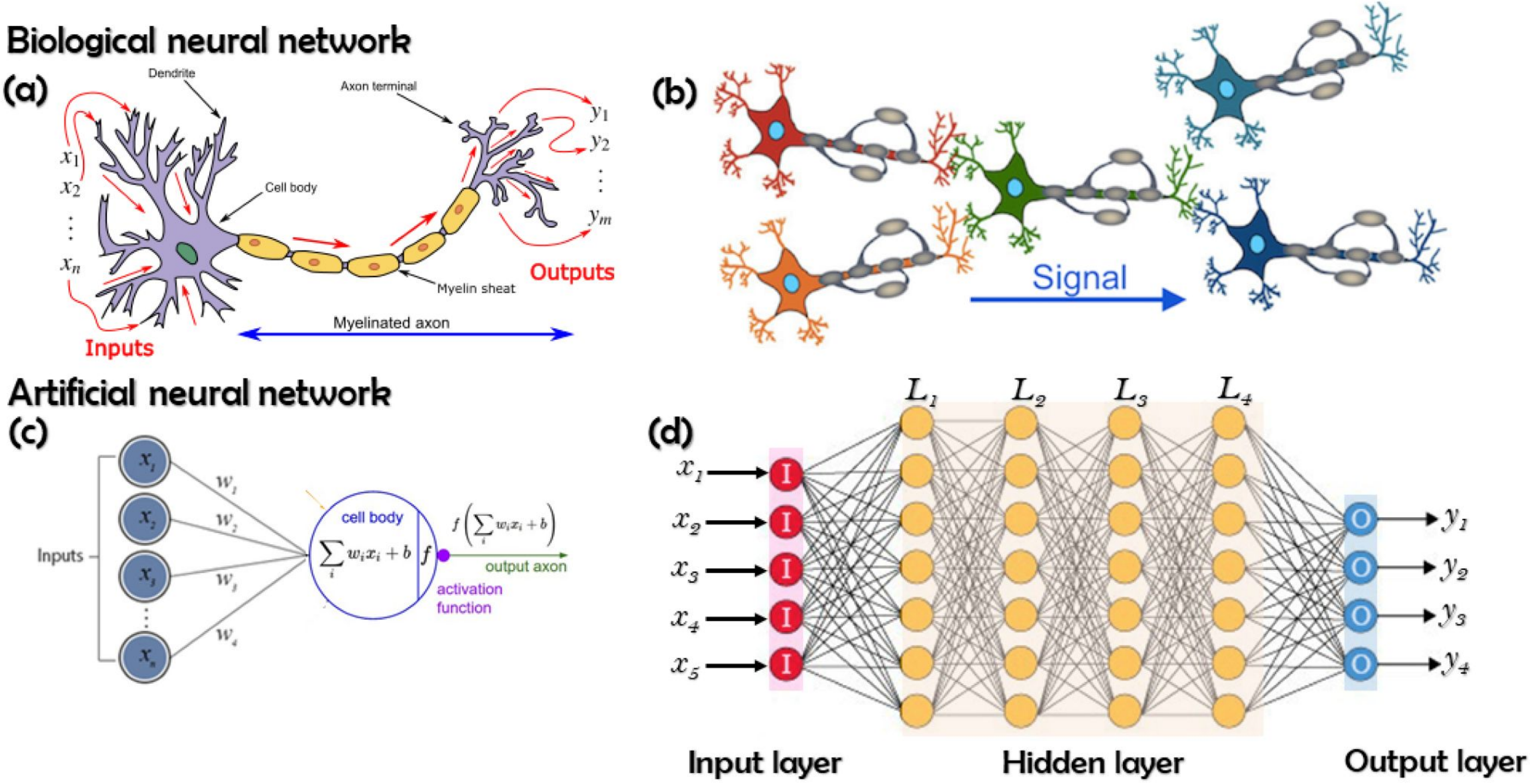
Data-Driven approach

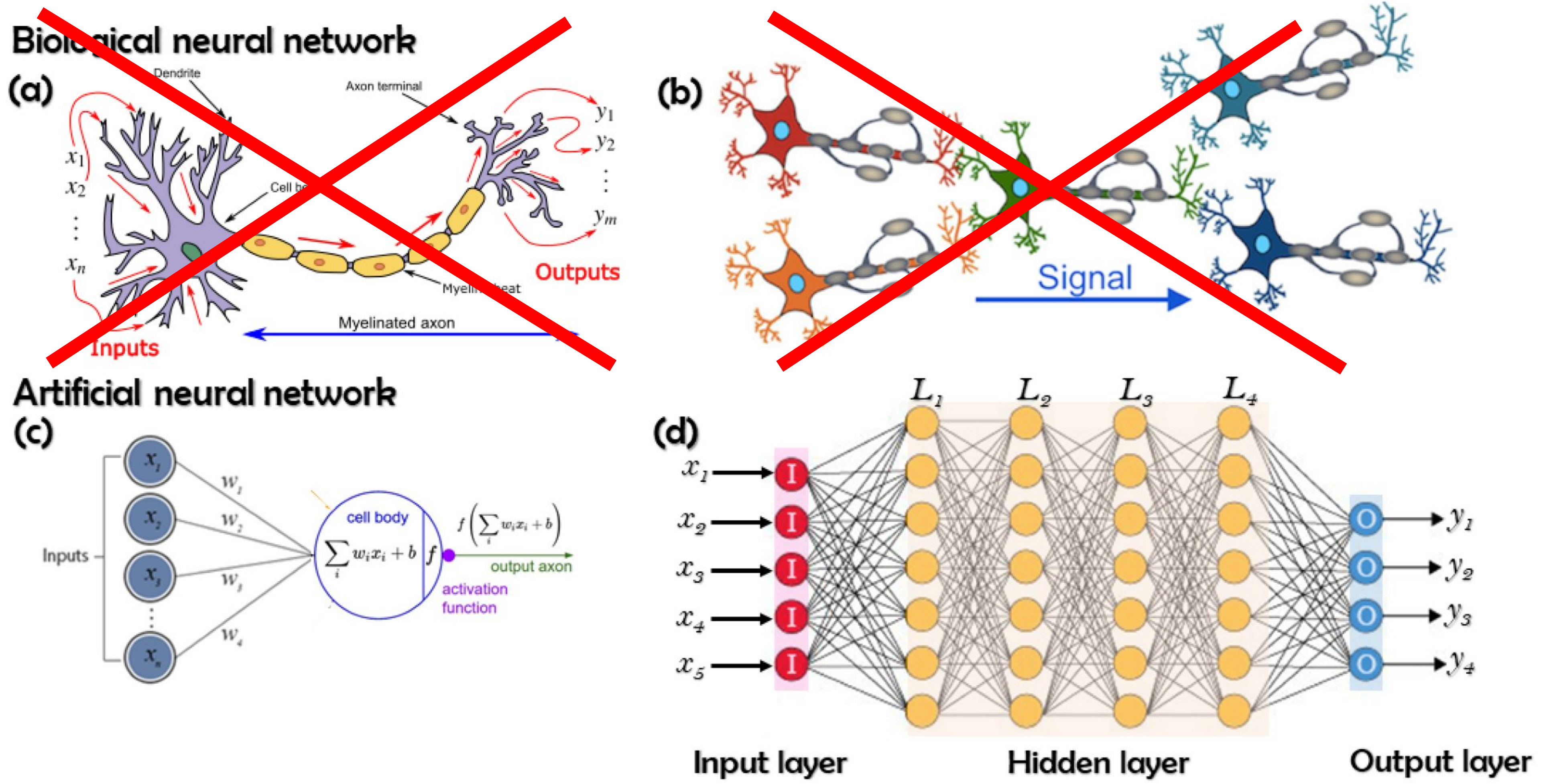
1. **Collect and label dataset**
2. **Train classifier**
3. **Predict on new images**



(Fully Connected) Neural Network

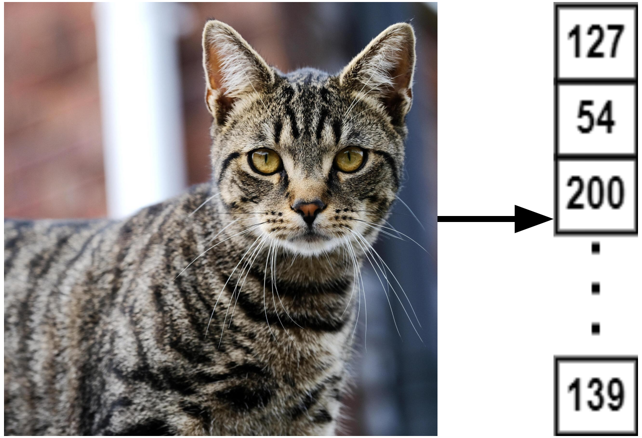


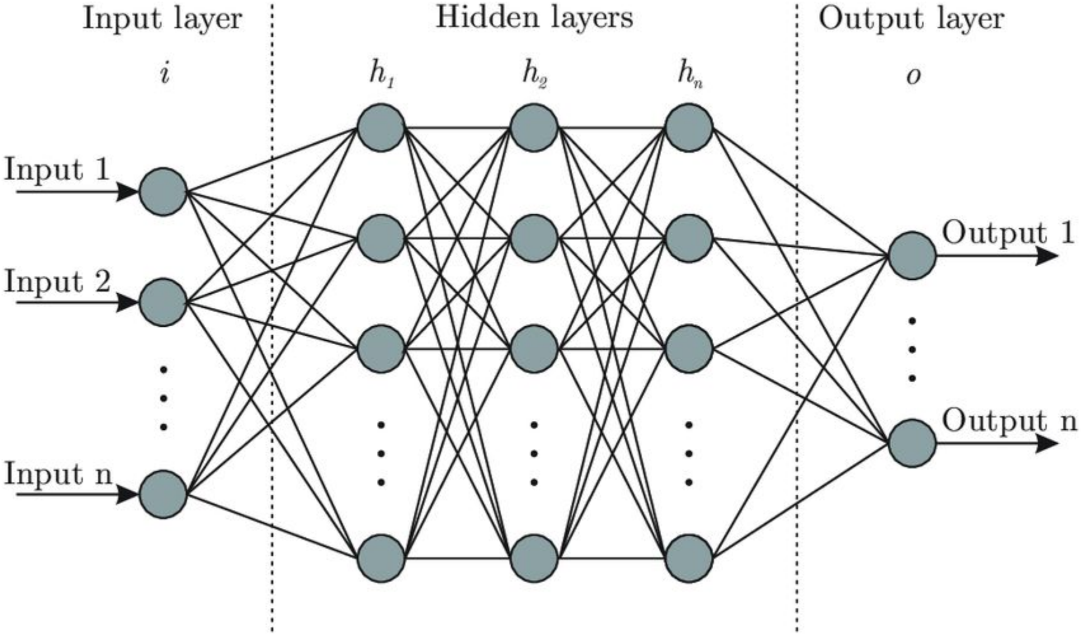

Biological neural network
(a)
Dendrite
Axon terminal
Cell body
Myelin sheat
Myelinated axon
Inputs
Outputs
x_1 x_2 x_n
y_1 y_2 y_m
(b)
Signal
Artificial neural network
(c)
Inputs
x_1 x_2 x_3 x_n
w_1 w_2 w_3 w_4
cell body
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
(d)
L_1 L_2 L_3 L_4
x_1 x_2 x_3 x_4 x_5
y_1 y_2 y_3 y_4
Input layer
Hidden layer
Output layer



Biological neural network
(a)
Dendrite
Axon terminal
x_1
x_2
x_n
y_1
y_2
y_m
Outputs
Myelinated axon
Inputs
(b)
Signal
Artificial neural network
(c)
Inputs
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_4
cell body
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
(d)
L_1
L_2
L_3
L_4
x_1
x_2
x_3
x_4
x_5
I
O
y_1
y_2
y_3
y_4
Input layer
Hidden layer
Output layer

Fully Connected Neural Network

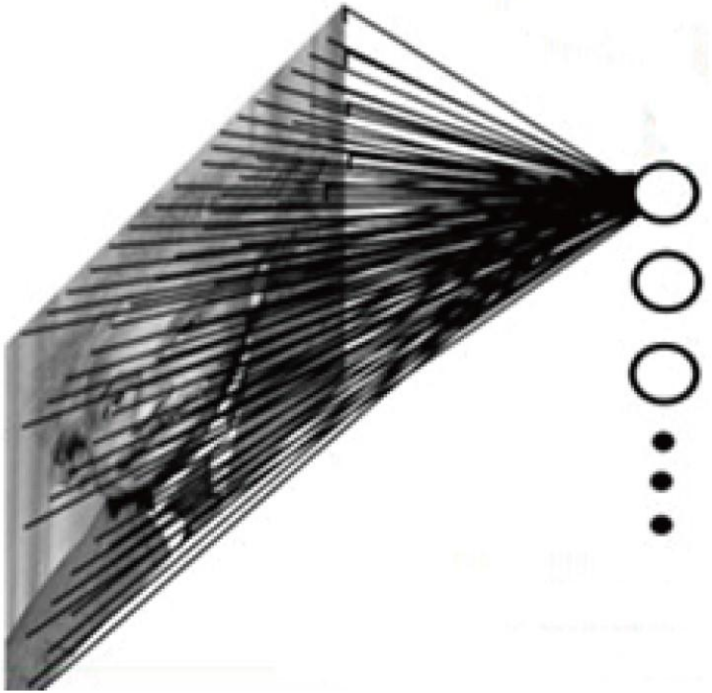




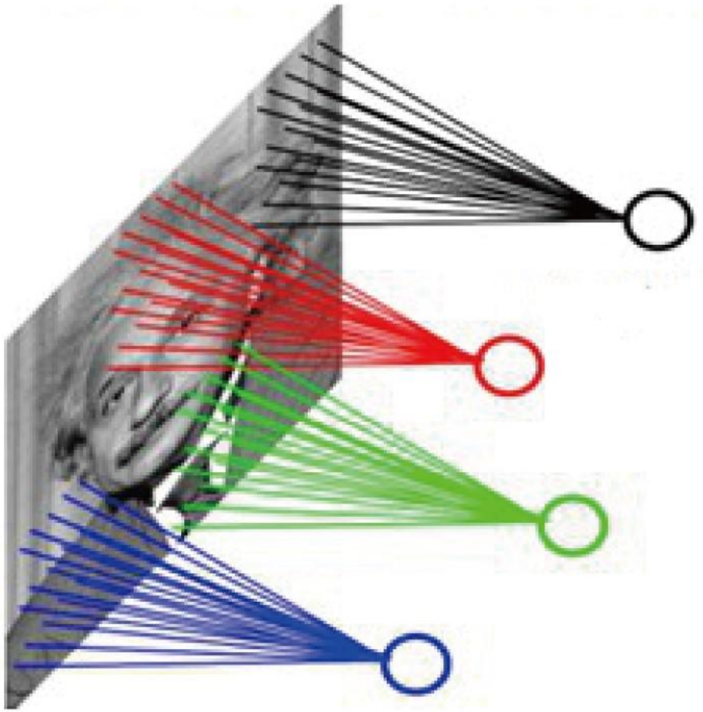
Problems:

- **Spatial information is lost**
- **A lot of parameters**

Using Spatial Structure

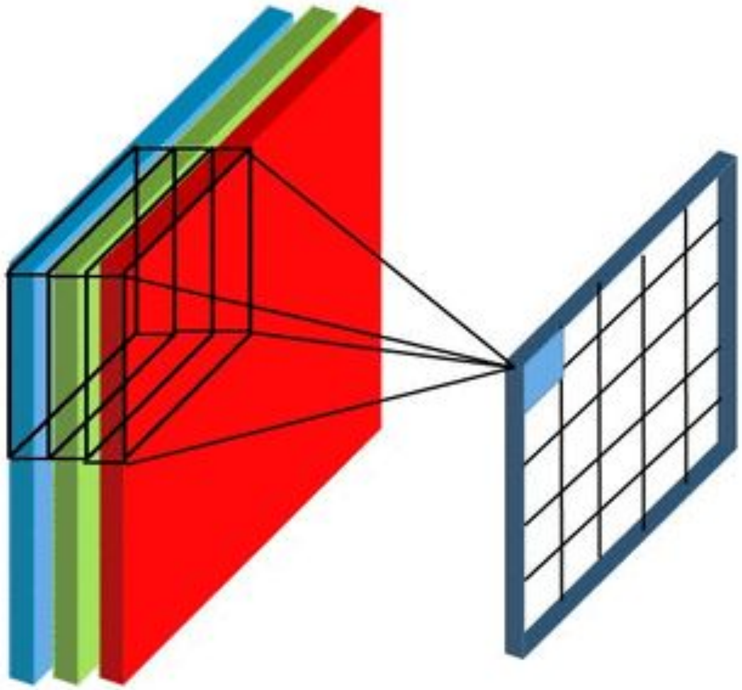


Fully connected

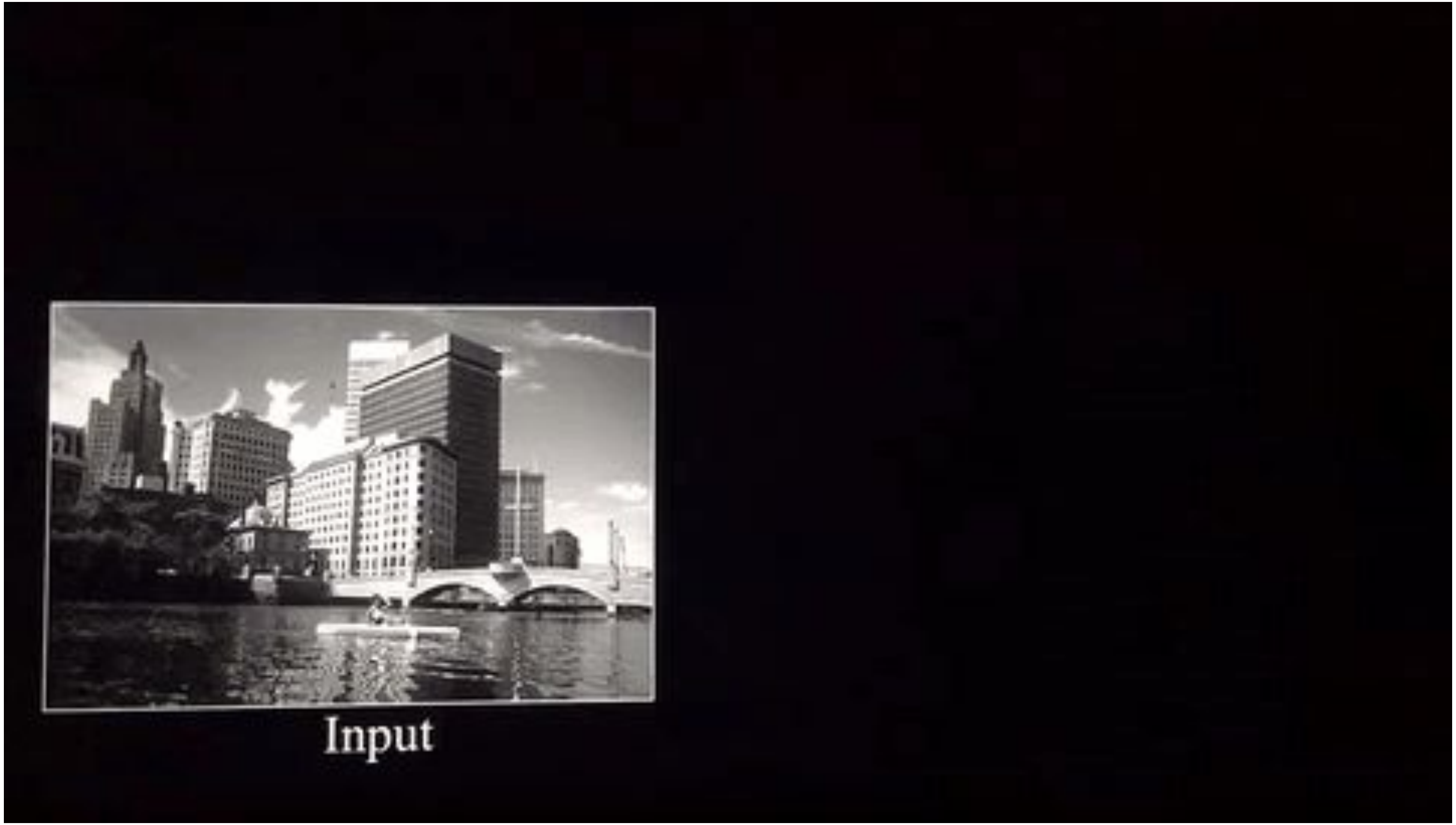


Locally connected

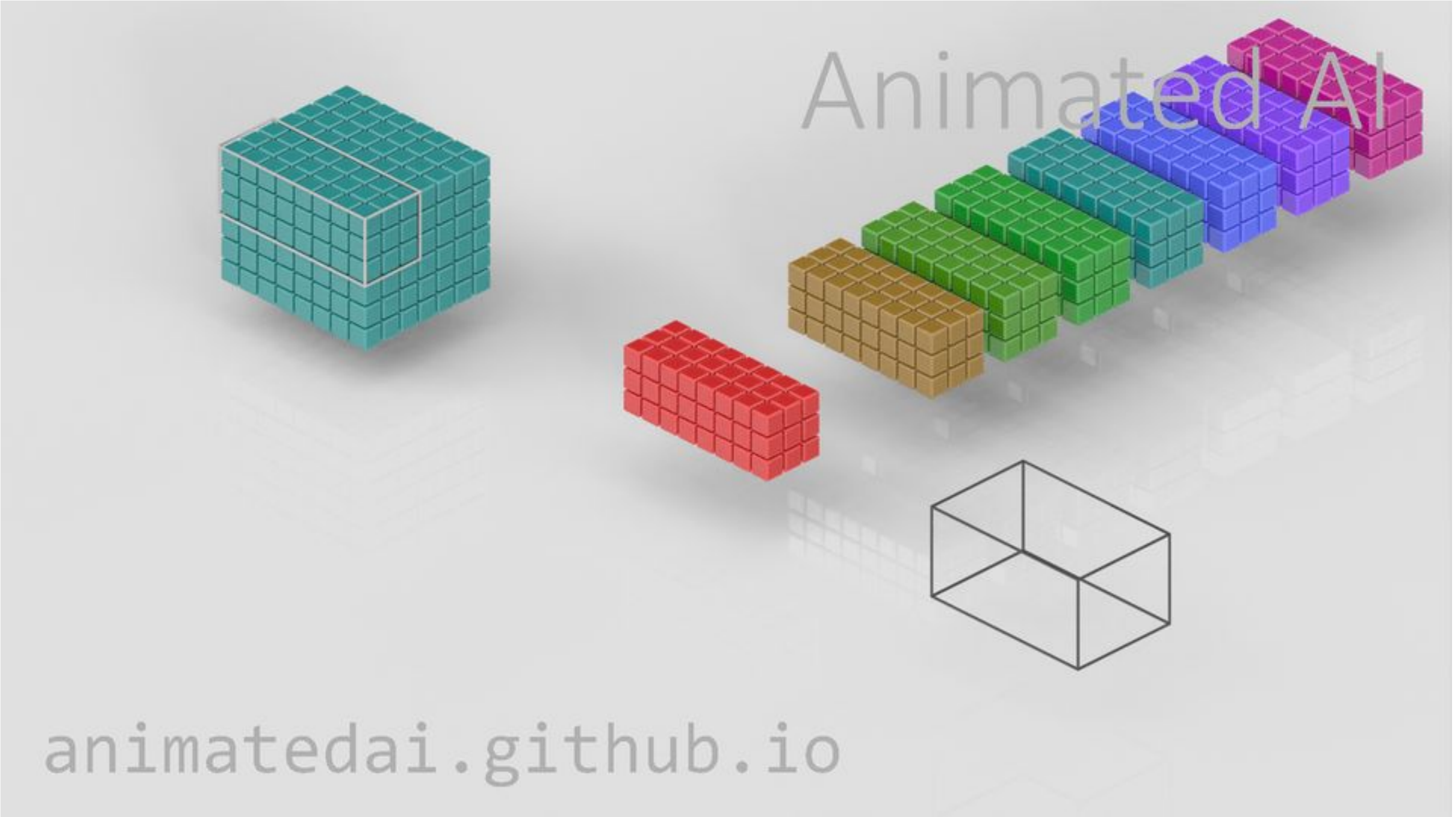
Convolution is applied to all channels of an image



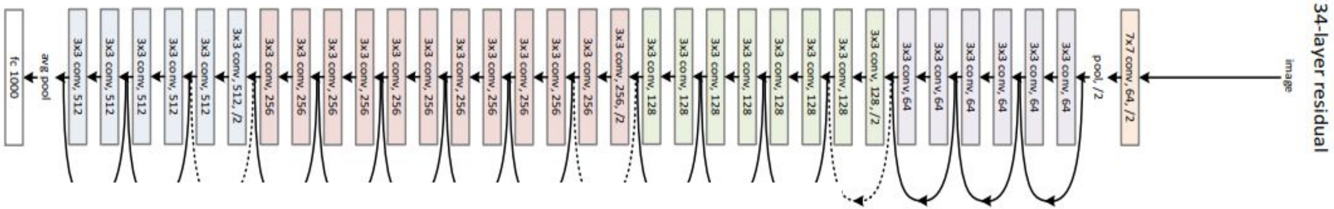
Convolution output

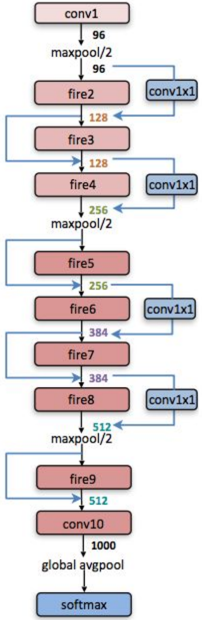


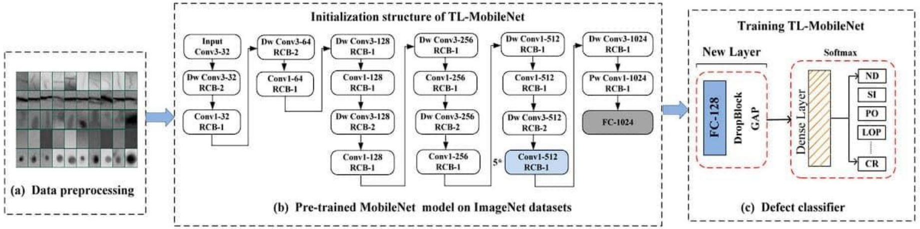
Convolution works with any number of channels



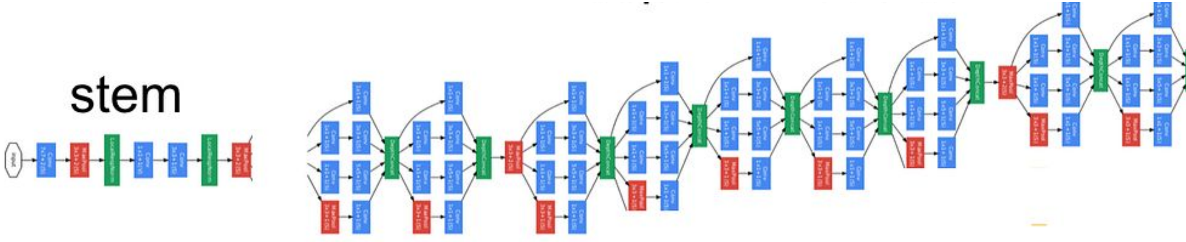
Modern CNN architectures (backbones)







stem



Computer Vision Tasks

Semantic Segmentation | **Classification + Localization** | **Object Detection** | **Instance Segmentation**

GRASS, CAT, TREE, SKY | CAT | DOG, DOG, CAT | DOG, DOG, CAT

No objects, just pixels | Single Object | Multiple Object

Semantic segmentation: Label each pixel with a class

SegNet

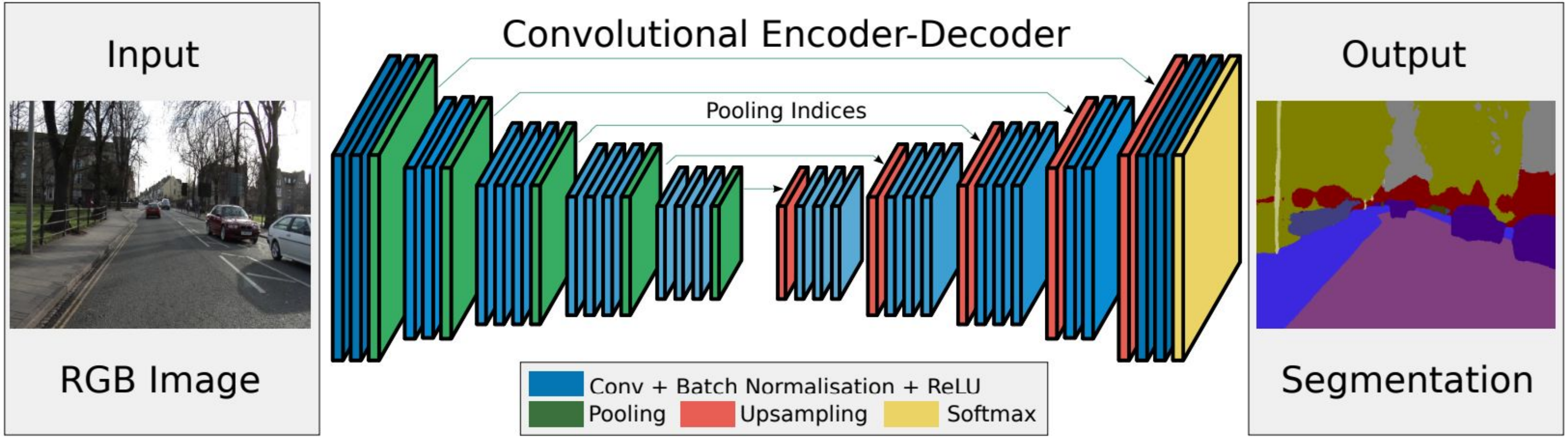
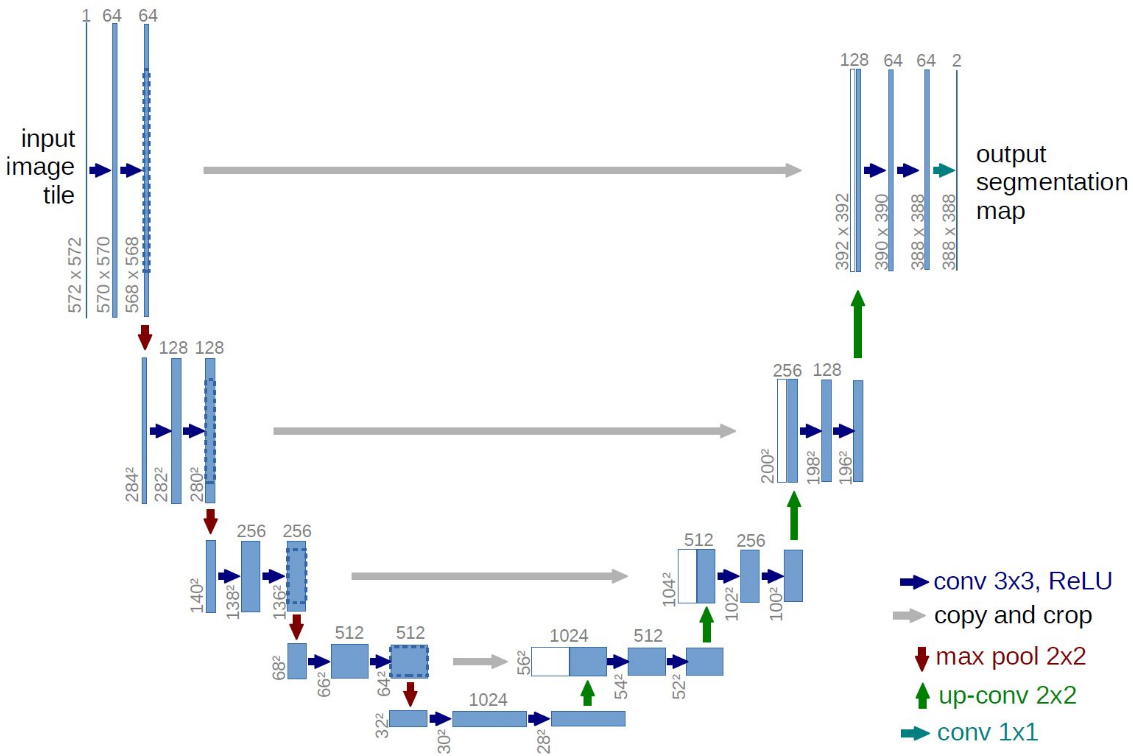


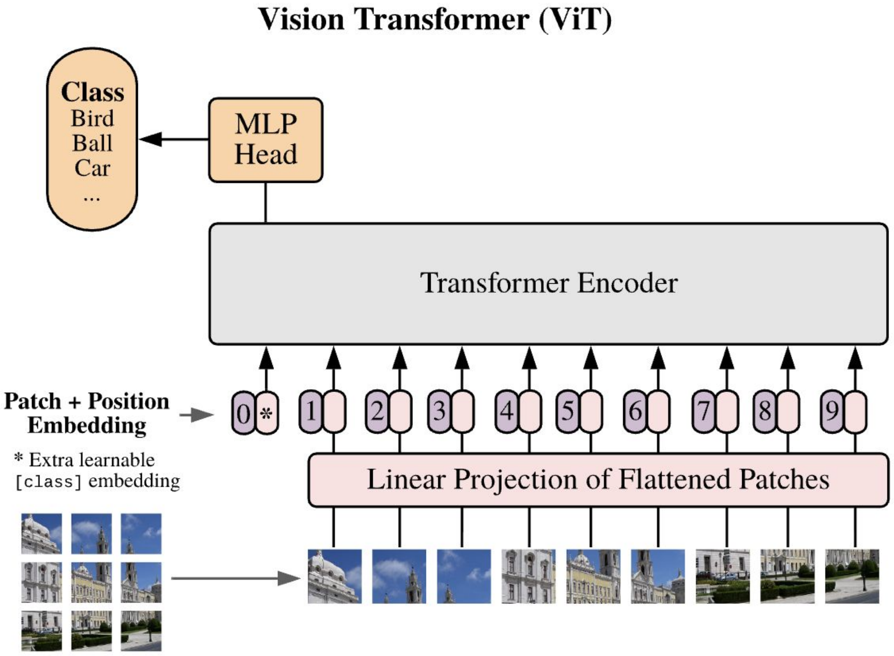
Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

U-Net: Convolutional Networks for Biomedical Image Segmentation



Vision Transformers (ViTs)

- Different than CNNs
- Split image into non-overlapping patches
- Process them like they are "words"
- O(n^2) complexity
    - n = number of patches
- Usually, can scale better with data than CNNs



Transformers are general token-processors

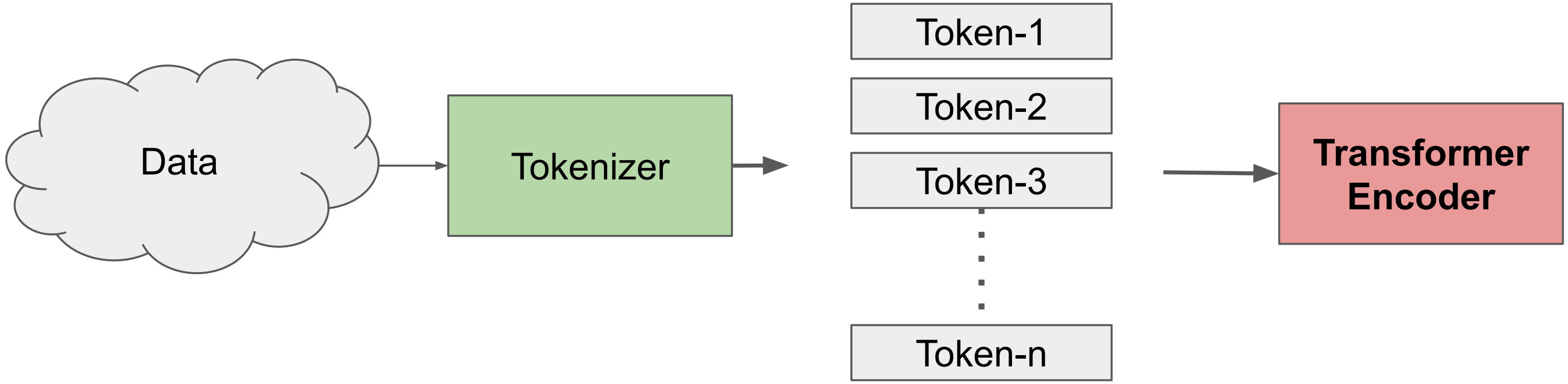
- Anything can be thought as a sequence of tokens
    - With some built-in order, or not

- **For text**: each character / subword / word is a token.
- **For images**: image patches? Pixels?
- **For videos:** ??
- **For sound waves:** ??

Transformers are general token-processors

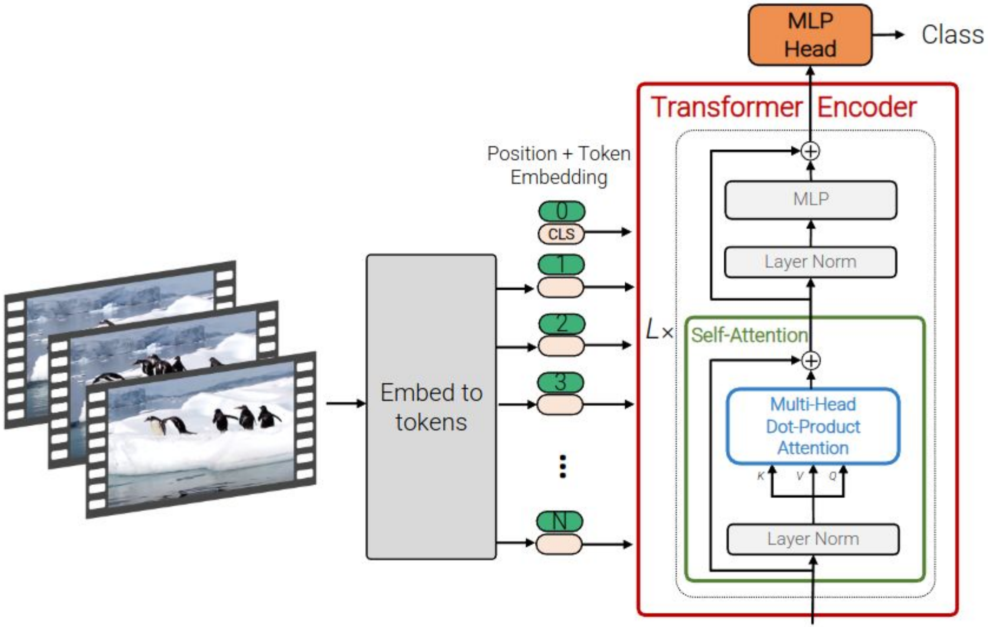
- Input data –> tokenizer –> transformer

- Most of the time, to solve a specific problem, we need to work on a custom tokenizer and position embeddings
    - e.g. how to tokenize videos / movement / audio?
    - e.g. how to encode time as positional embedding? How to handle multi-modal data?



ViViT: A Video Vision Transformer

How would a tokenizer look for videos?



ViViT: A Video Vision Transformer

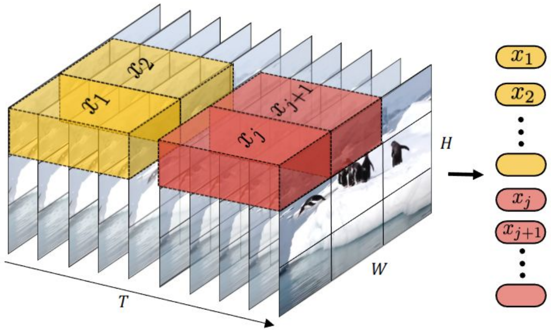
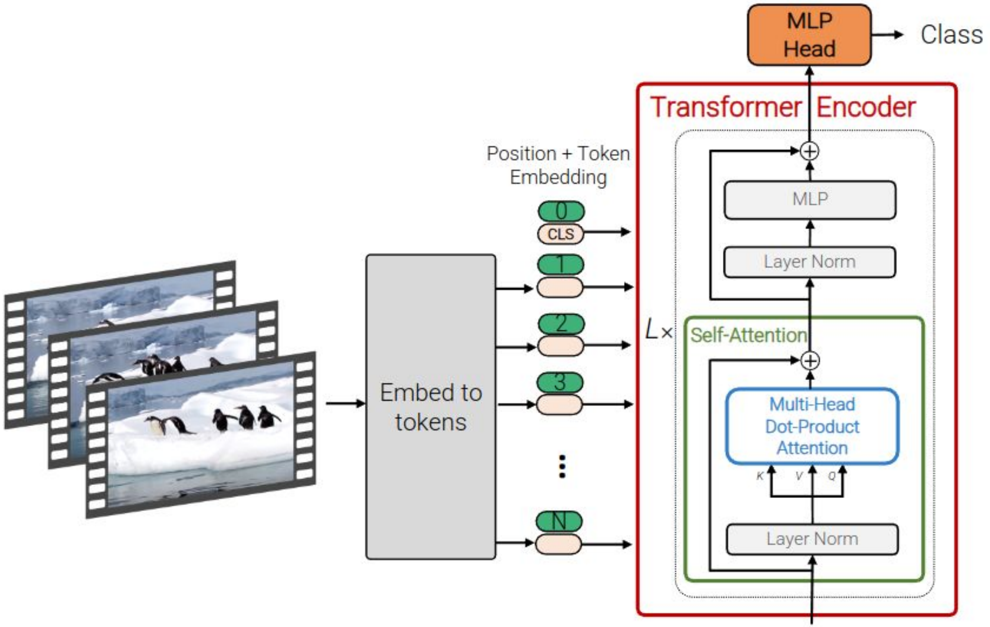


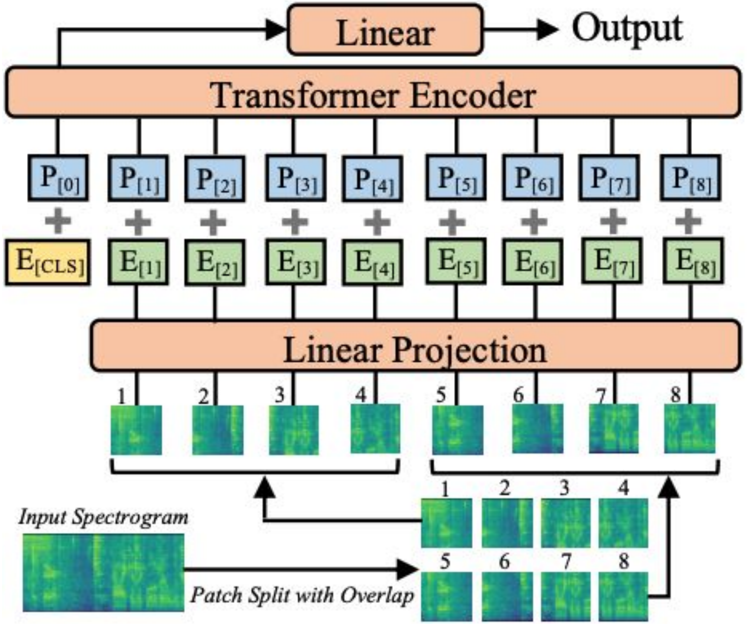
Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.



Audio Spectrogram Transformer

- Transform audio waves into spectrograms

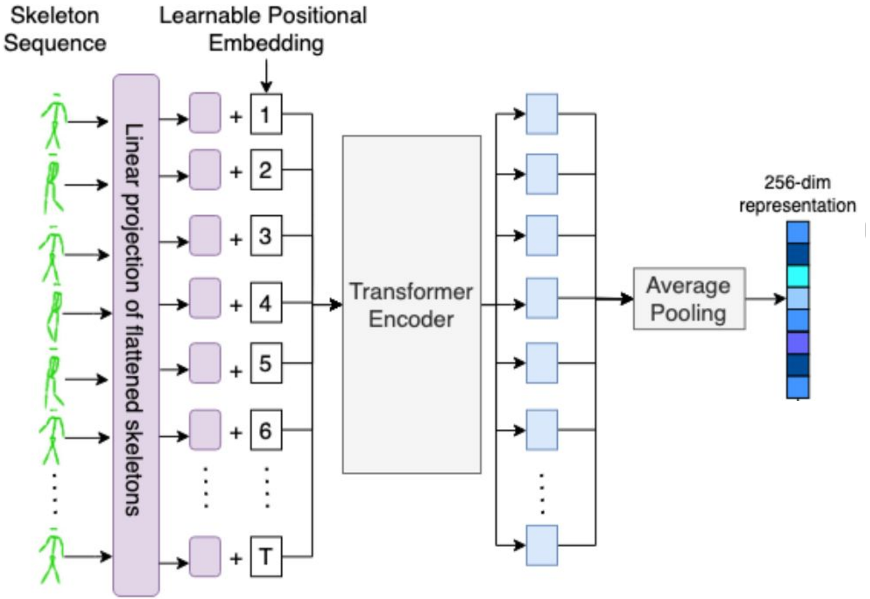
- Spectrograms kind of look like images
  - Let's just use the same thing as in ViTs
  - Patchify and send to Transformer encoder



GaitFormer (Cosma and Rădoi, 2022)

- How can we process skeleton sequences?
- A skeleton sequence is represented by a Tx18x3 matrix
  - T frames, 18 joints, 3 coordinates

- Basic idea:
  - Flatten each skeleton and send the sequence to the transformer encoder



When to pick what?

- ViTs have better global reasoning and multi-tasking
    - Might not be so appropriate
    - O(n^2) time and memory complexity

- CNN are fast and work best for edge
- Many optimized variants
    - MobileNet, SqueezeNet, ConvNext-tiny, etc.

A ConvNet for the 2020s

Zhuang Liu[1,2*] Hanzi Mao[1] Chao-Yuan Wu[1] Christoph Feichtenhofer[1] Trevor Darrell[2] Saining Xie[1†]

[1]Facebook AI Research (FAIR) [2]UC Berkeley

Code: https://github.com/facebookresearch/ConvNeXt

Abstract

The "Roaring 20s" of visual recognition began with the introduction of Vision Transformers (ViTs), which quickly superseded ConvNets as the state-of-the-art image classification model. A vanilla ViT, on the other hand, faces difficulties when applied to general computer vision tasks such as object detection and semantic segmentation. It is the hierarchical Transformers (e.g., Swin Transformers) that reintroduced several ConvNet priors, making Transformers practically viable as a generic vision backbone and demonstrating remarkable performance on a wide variety of vision tasks. However, the effectiveness of such hybrid approaches is still largely credited to the intrinsic superiority of Transformers, rather than the inherent inductive biases of convolutions. In this

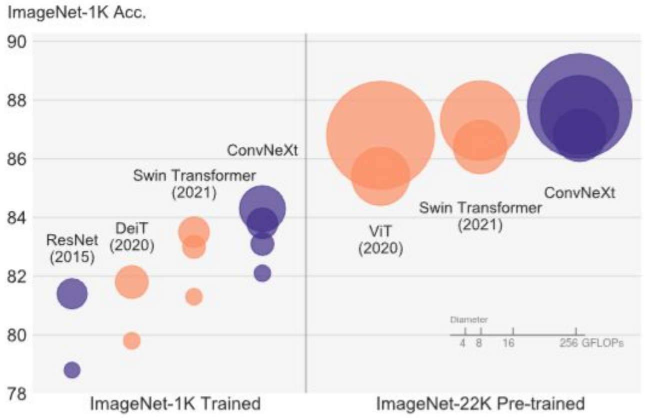
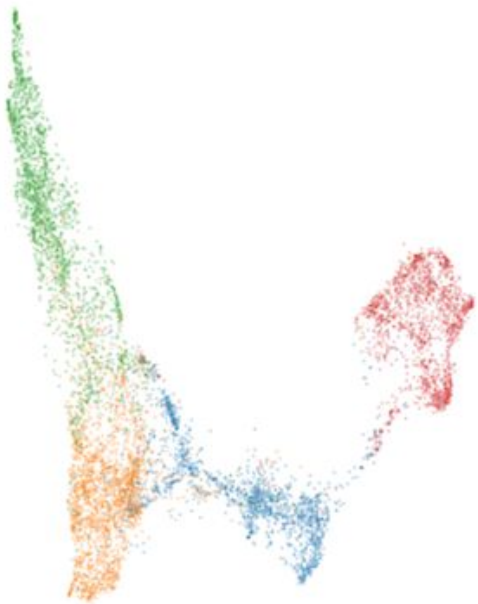
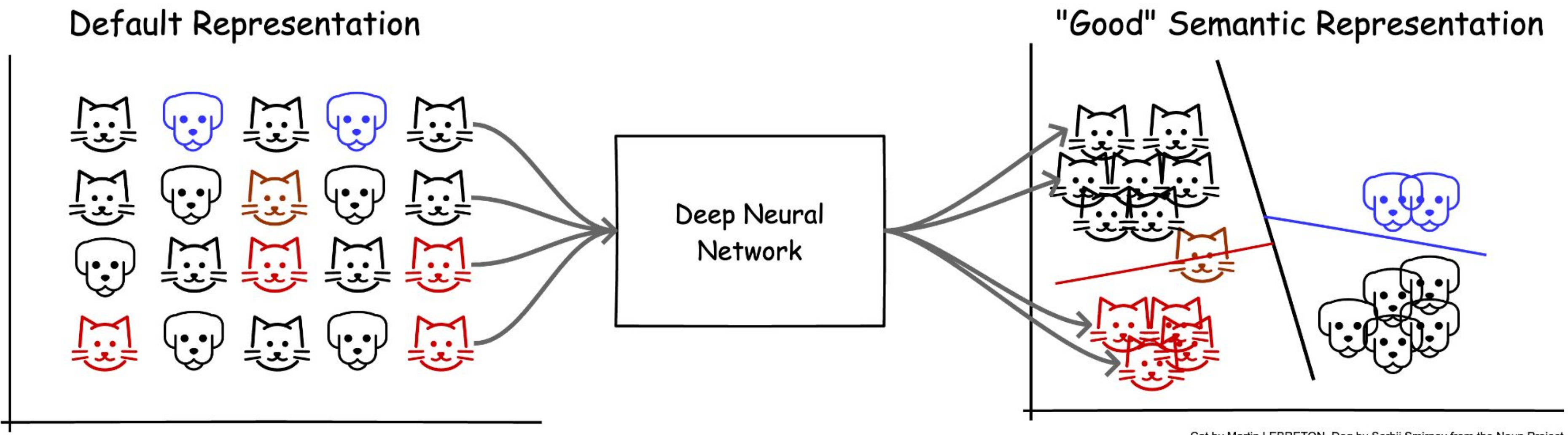


Figure 1. **ImageNet-1K classification** results for • ConvNets and

Let’s talk about “embeddings”



Deep Learning = Representation Learning



Cat by Martin LEBRETON, Dog by Serhii Smirnov from the Noun Project

Semantic Representation = Similarity

f( , ) =

Semantic Representation = Similarity

f( , ) = $\uparrow$

f( , ) = $\downarrow$

Embeddings are like hashes

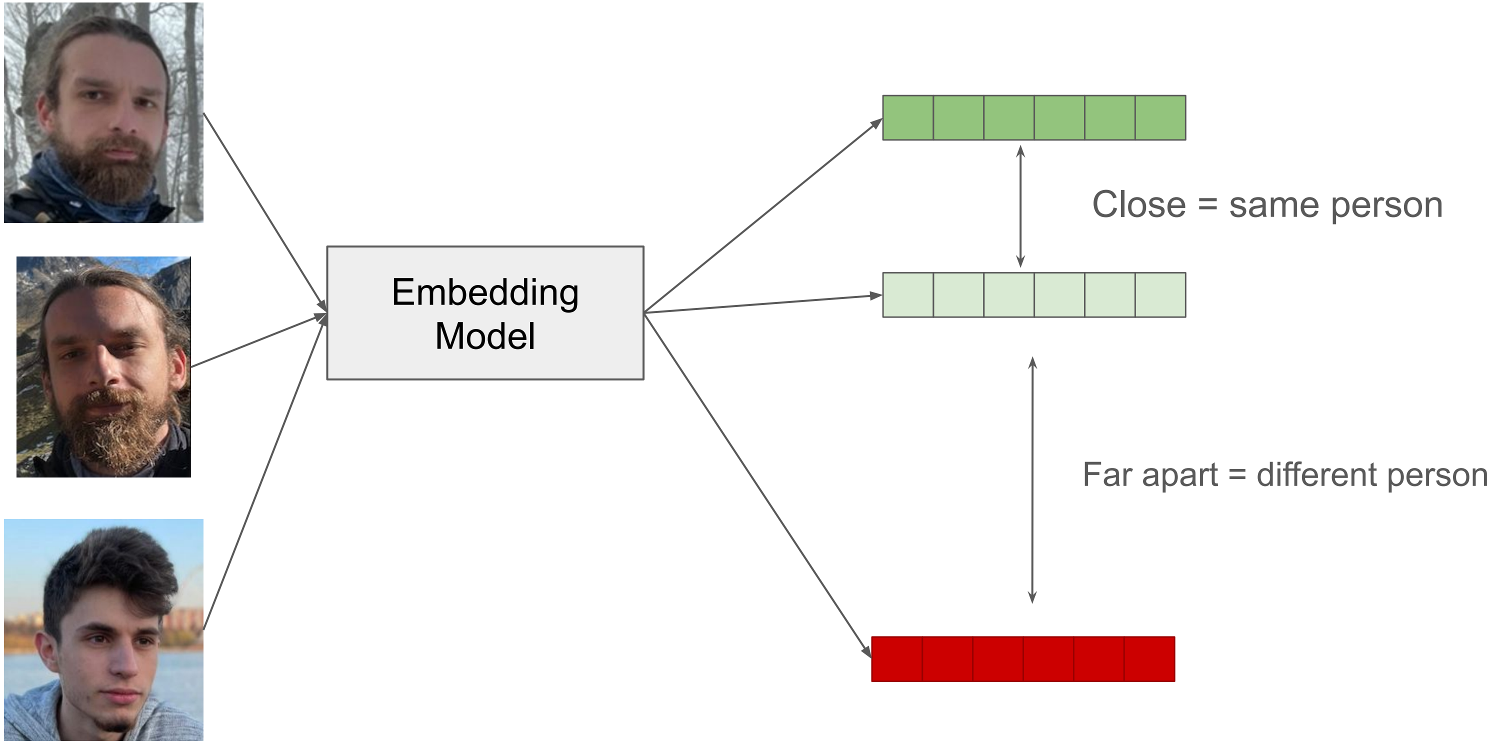
- Two hashes are near each other means the semantic content is the same

- The embeddings of two pictures of cats should be similar, regardless of surface level differences

- Really, an embedding is just a vector of floats with certain properties
    - Usually 64 - 4096 dimensional vector (depending on the model)

1.3	0.3	1.1	0.9	0.8	0.7	0.1	0.4	0.8	1.5	1.6	2.3	4.2

Face Authentication with face embeddings

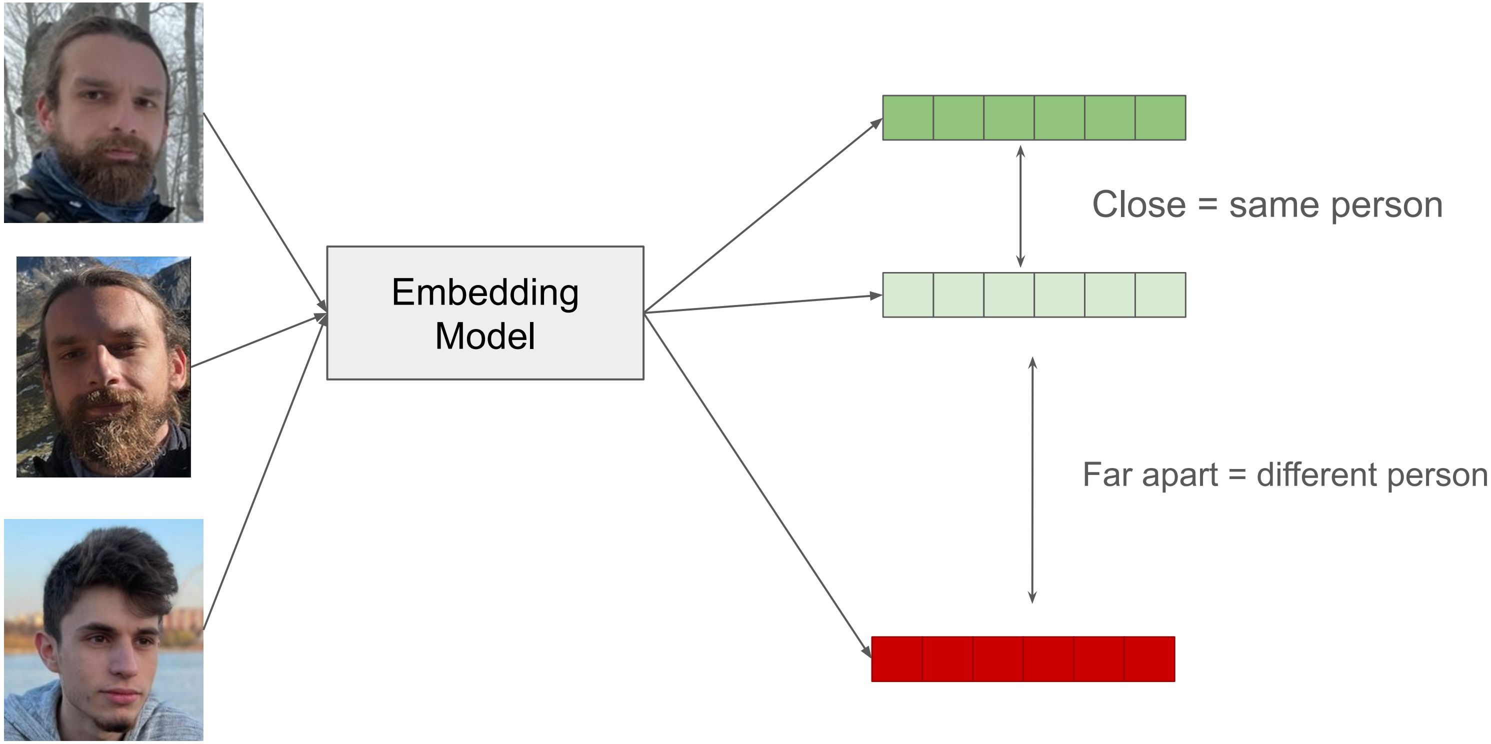


“Closeness” = dot product similarity

$$\mathbf{a} = \langle a_1, a_2, a_3 \rangle \quad \mathbf{b} = \langle b_1, b_2, b_3 \rangle$$

$$\boxed{\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3}$$

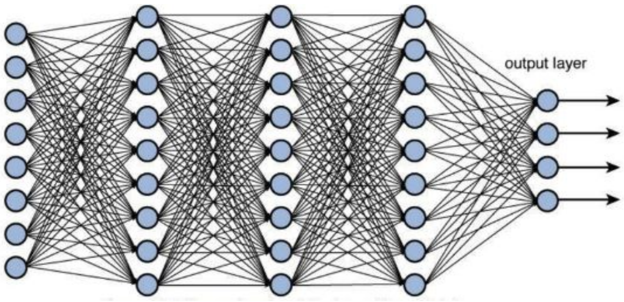
Face Authentication with face embeddings



How are these embeddings learned by a model?

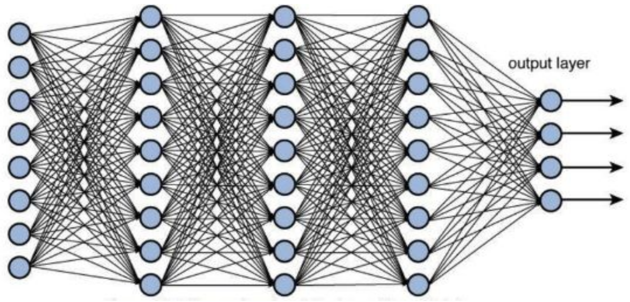
Components of an ML Application

Model (Neural Network) + Training Data + Hardware (GPU / TPU)



Components of an ML Application

Model (Neural Network)



+

Training Data

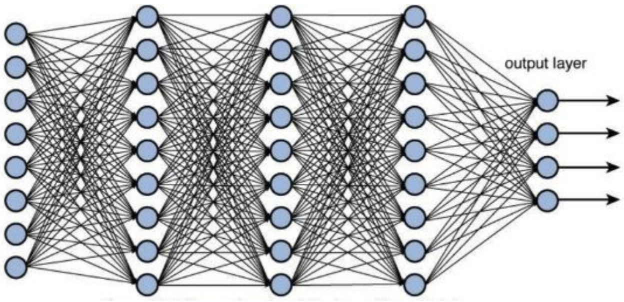
+

Hardware (GPU / TPU)

Fancy DL Architectures
Available (timm, torchvision, 🤗HuggingFace etc.)

Components of an ML Application

Model (Neural Network)



+

Training Data

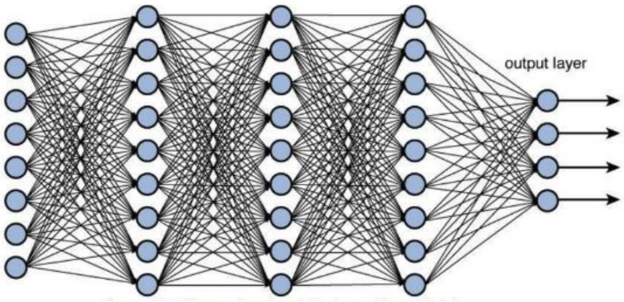
+

Hardware (GPU / TPU)

Usually available
Google Colab / cloud, gaming rigs, university cluster etc.

Components of an ML Application

Model (Neural Network)



+

Training Data

+

Hardware (GPU / TPU)

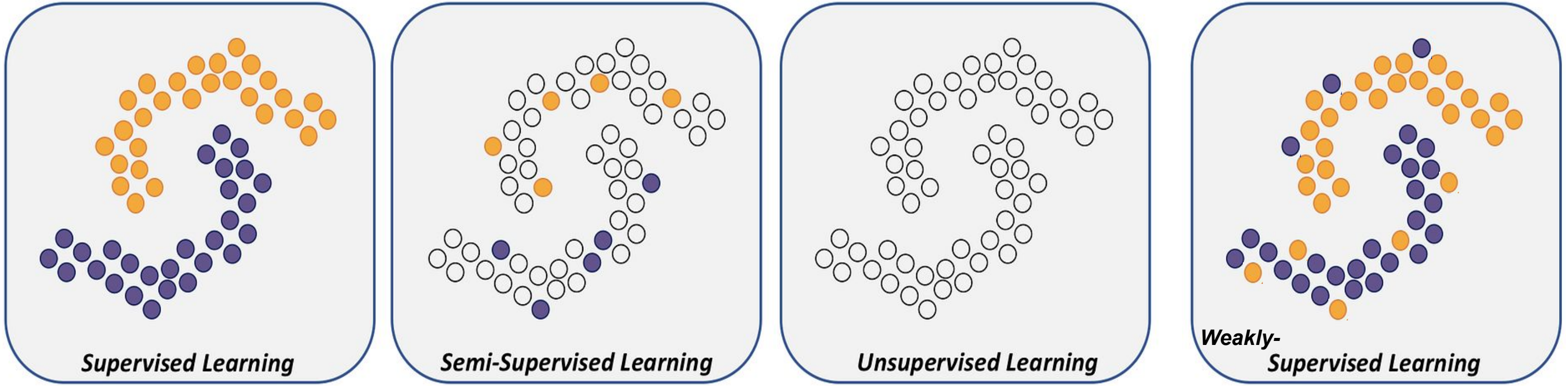
?????????

The **most important** aspect
This is where **supervision** is performed

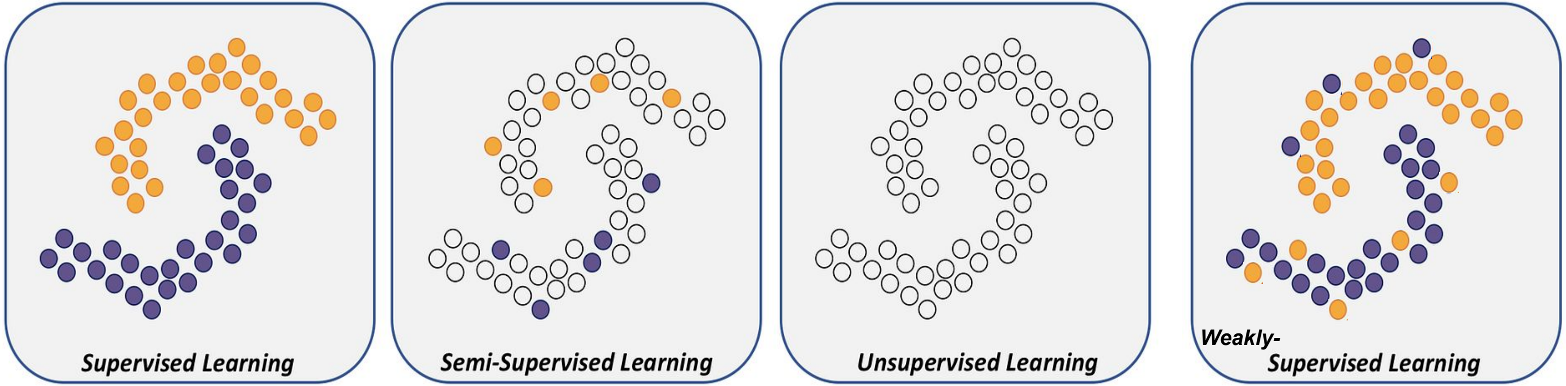
Model differences are ***<u>overrated</u>***

Supervision differences are ***<u>underrated</u>***.

Types of “supervision”



Types of “supervision”



Ideal world! Not really happening

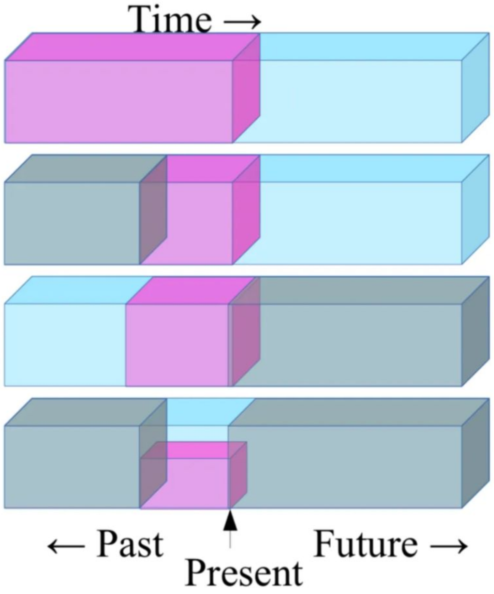
Closer to the real world

Yann LeCun

Y. LeCun

Self-Supervised Learning

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.



Pretext-based Self-Supervised Learning

Main Idea:

- Invent a task from the data and force the model to solve it

- Solving the task = understanding the data

How is this image rotated?

How is this image rotated?

How do you know?

Self-Supervised Learning: Predicting Rotations

Predict rotations

0 degrees

270 degrees

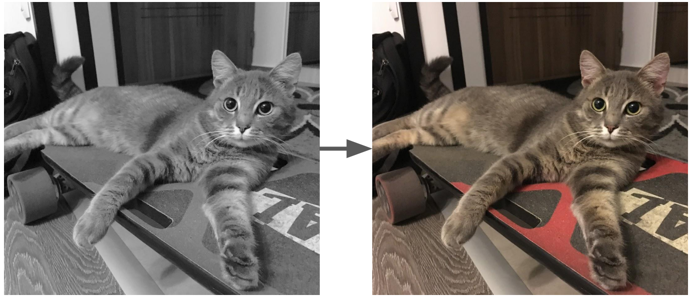
180 degrees

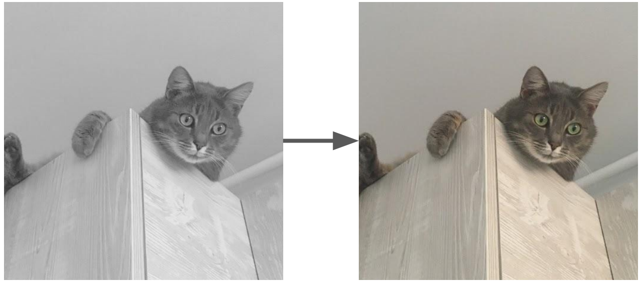
90 degrees

Hypothesis: a model could recognize the correct rotation of an object only if it has the “visual commonsense” of what the object should look like unperturbed.

Self-Supervised Learning: Image Colorization

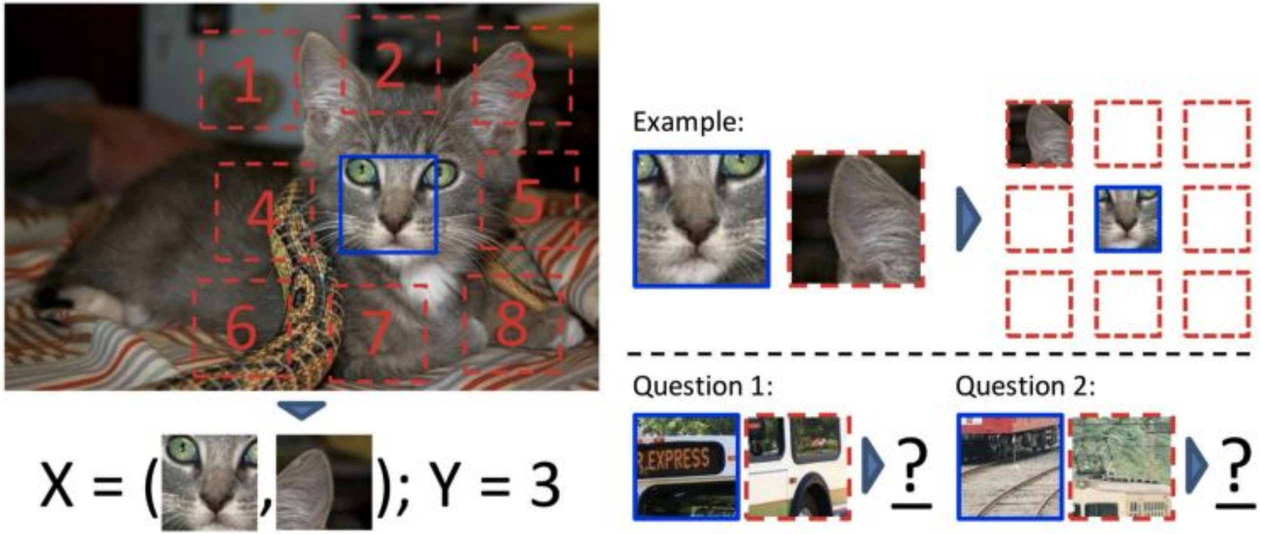
Image Colorization





Hypothesis: a model could only colorize an image if it has the “visual commonsense” of what the object should look like.

Self-Supervised Learning: Predicting Relative Patches



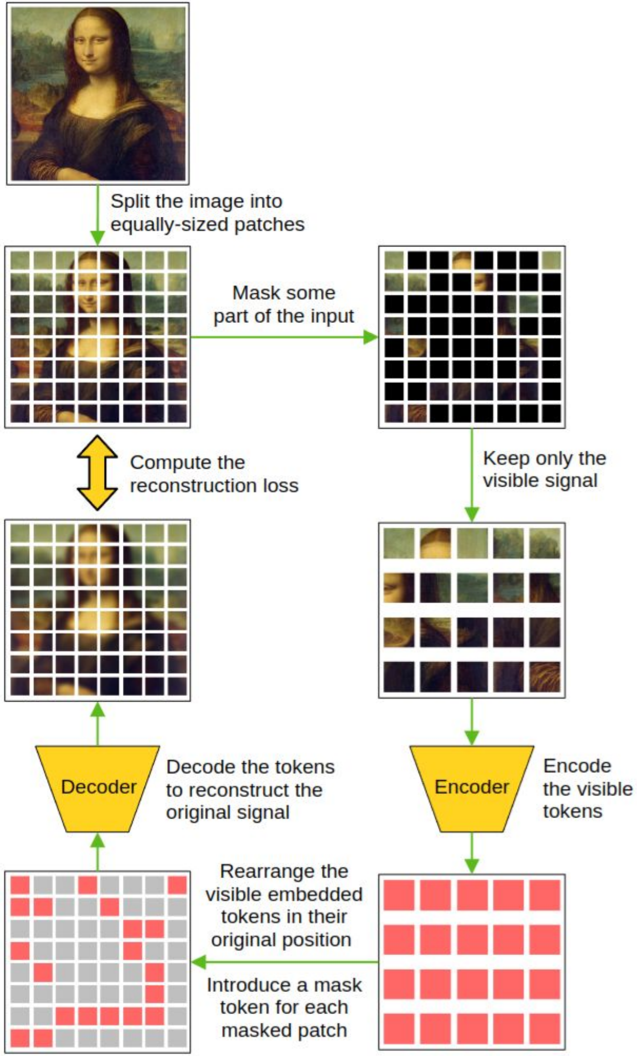
(Image source: Doersch et al., 2015)

Hypothesis: a model could only predict relative patches from an image if it has the "visual commonsense" of what the global object should look like.

Masked-Image Modelling

- Mask some parts of the image

- Predict the masked parts given the visible parts



Models can cheat!

i.e. find shortcuts to solve the task, don’t understand the overall semantics of the data

Spurious correlations

Published as a conference paper at ICLR 2019

IMAGENET-TRAINED CNNS ARE BIASED TOWARDS TEXTURE; INCREASING SHAPE BIAS IMPROVES ACCURACY AND ROBUSTNESS

Robert Geirhos
University of Tübingen & IMPRS-IS
robert.geirhos@bethgelab.org

Patricia Rubisch
University of Tübingen & U. of Edinburgh
p.rubisch@sms.ed.ac.uk

Claudio Michaelis
University of Tübingen & IMPRS-IS
claudio.michaelis@bethgelab.org

Matthias Bethge*
University of Tübingen
matthias.bethge@bethgelab.org

Felix A. Wichmann*
University of Tübingen
felix.wichmann@uni-tuebingen.de

Wieland Brendel*
University of Tübingen
wieland.brendel@bethgelab.org

ABSTRACT

Convolutional Neural Networks (CNNs) are commonly thought to recognise objects by learning increasingly complex representations of object shapes. Some recent studies suggest a more important role of image textures. We here put these conflicting hypotheses to a quantitative test by evaluating CNNs and human observers on images with a texture-shape cue conflict. We show that ImageNet-

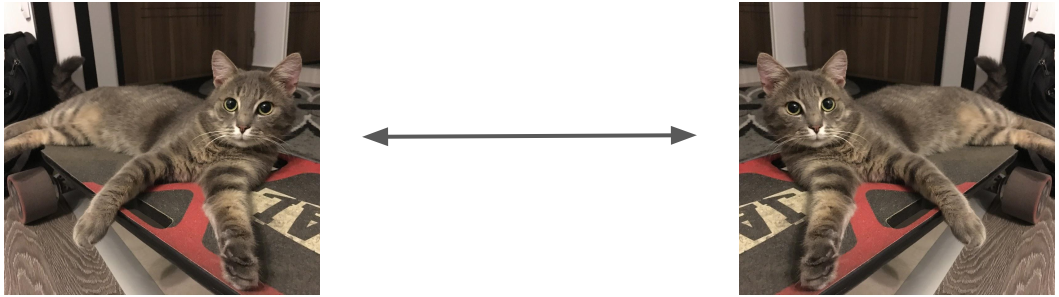
Example: CNNs are biased towards texture

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Question: Is it a good pretext-task to try to predict whether an image is mirrored or not?



Question: Which images are mirrored?

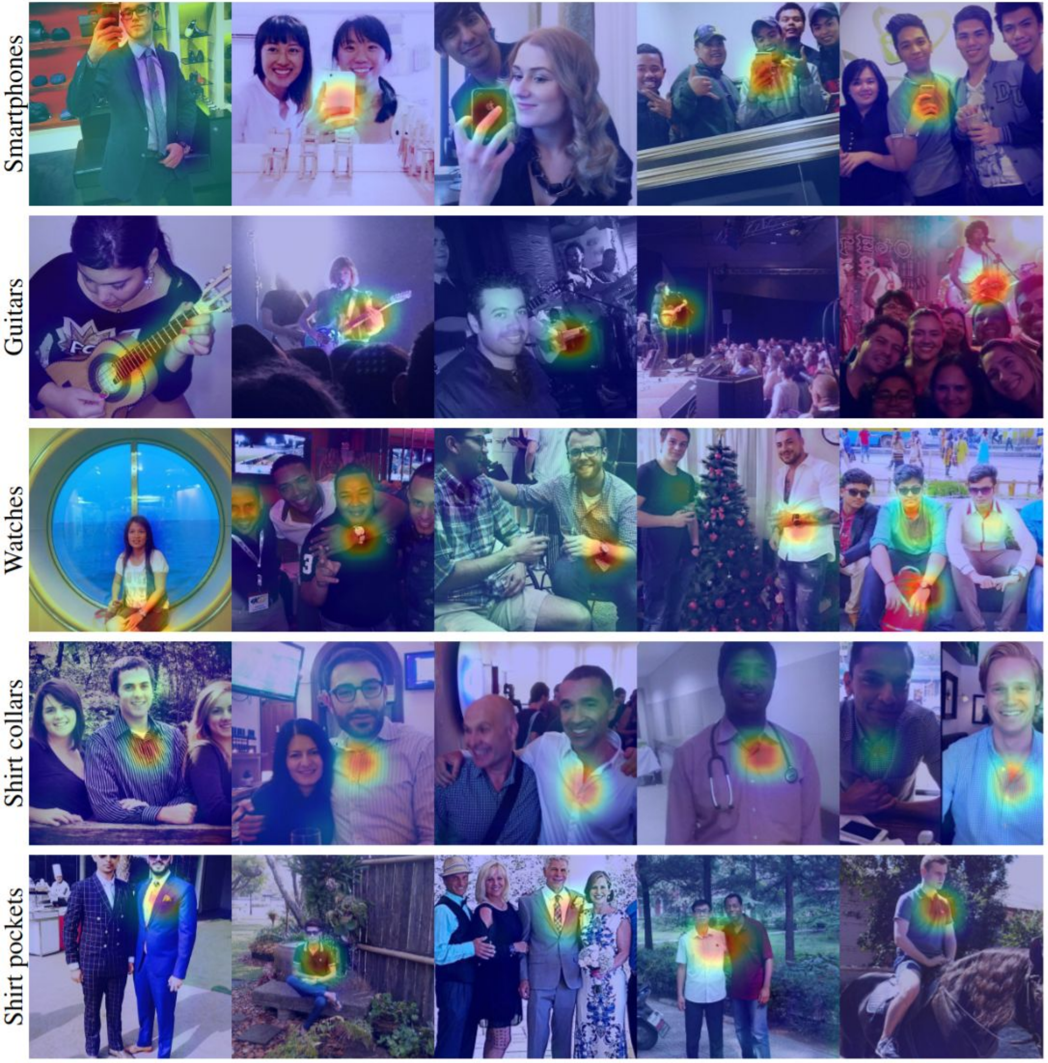
(a)

(b)

(c)

Question: Which images are mirrored?

(a) (b) (c)



Visual Chirality

Zhiqiu Lin[1] Jin Sun[1,2] Abe Davis[1,2] Noah Snavely[1,2]
Cornell University[1] Cornell Tech[2]

Abstract

How can we tell whether an image has been mirrored? While we understand the geometry of mirror reflections very well, less has been said about how it affects distributions of imagery at scale, despite widespread use for data augmentation in computer vision. In this paper, we investigate how the statistics of visual data are changed by reflection. We refer to these changes as "visual chirality," after the concept of geometric chirality—the notion of objects that are distinct from their mirror image. Our analysis of visual chirality reveals surprising results, including low-level chiral signals pervading imagery stemming from image processing in cameras, to the ability to discover visual chirality in images of people and faces. Our work has implications for data augmentation, self-supervised learning, and image forensics.

(a) (b) (c)

Figure 1. **Which images have been mirrored?** Our goal is to understand how distributions of natural images differ from their reflections. Each of the images here appears plausible, but some subset have actually been flipped horizontally. Figuring out which can be a challenging task even for humans. Can you tell which are flipped? Answers are in Figure 2.

less about how it changes what we learn from that data—this, despite widespread use of image reflection (e.g., mirror-flips)

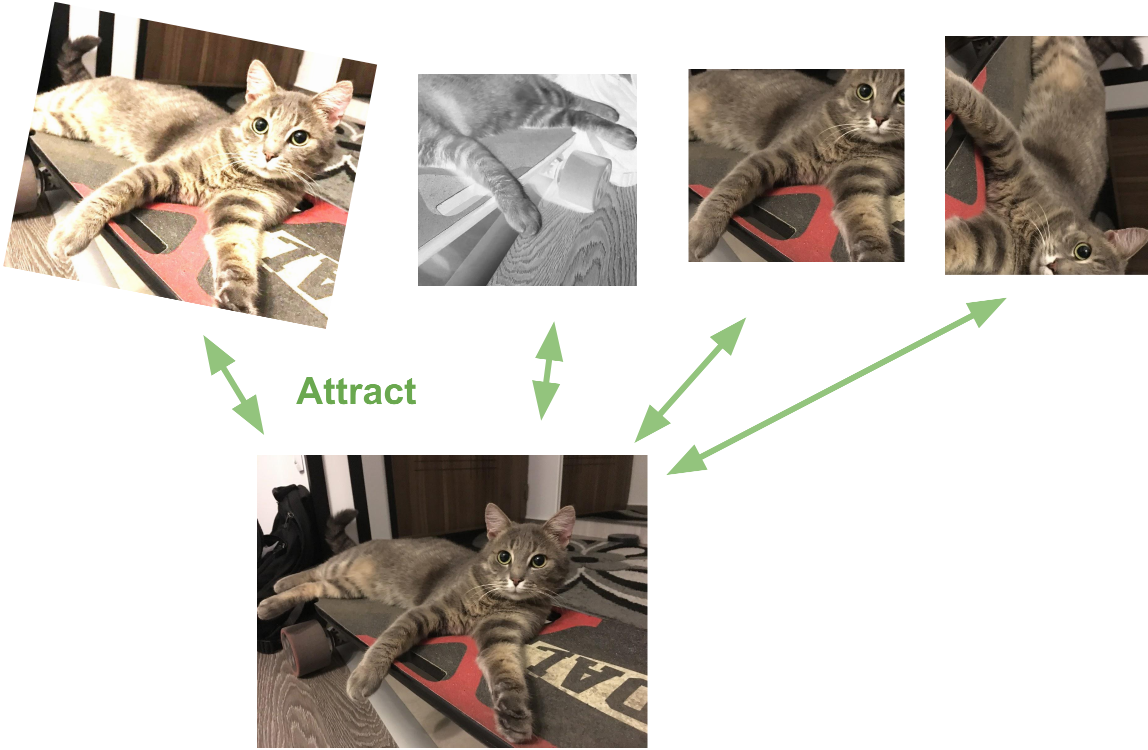
Learned representations may be tied to a specific pretext task!

Can we come up with a more general pretext task?

Contrastive Learning

Main Idea:

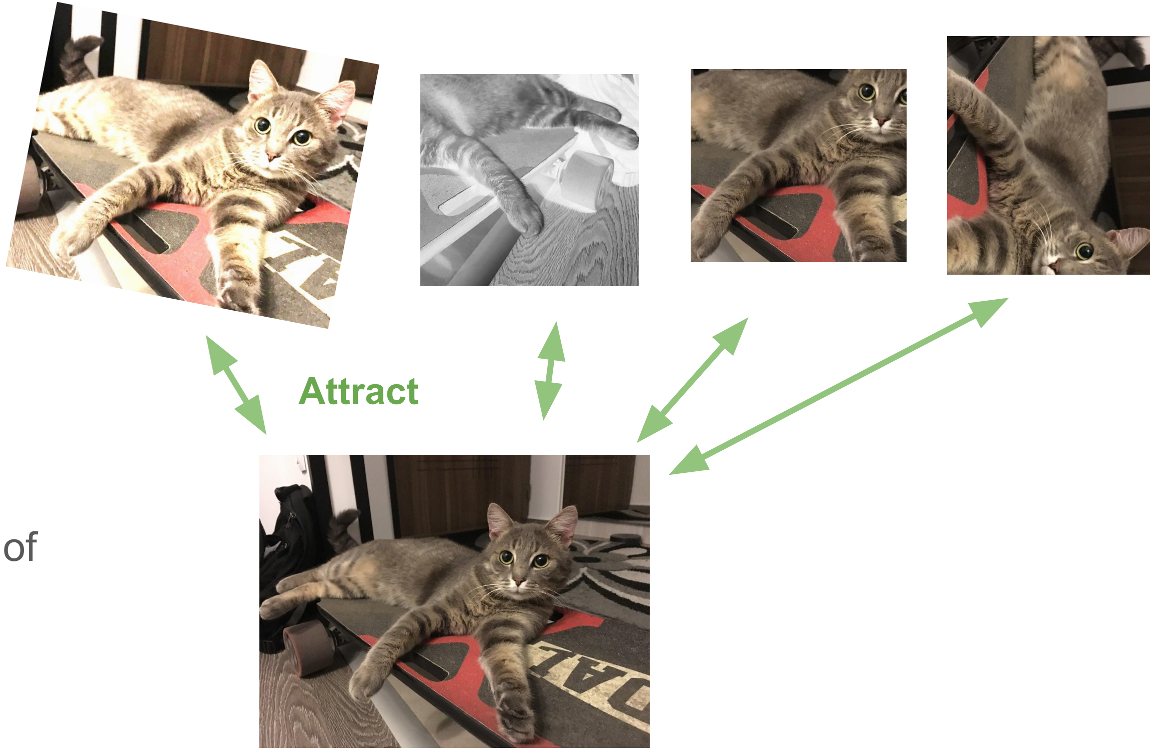
Multiple views of the same image should have the same representations



Contrastive Learning

Main Idea:

Multiple views of the same image should have the same representations

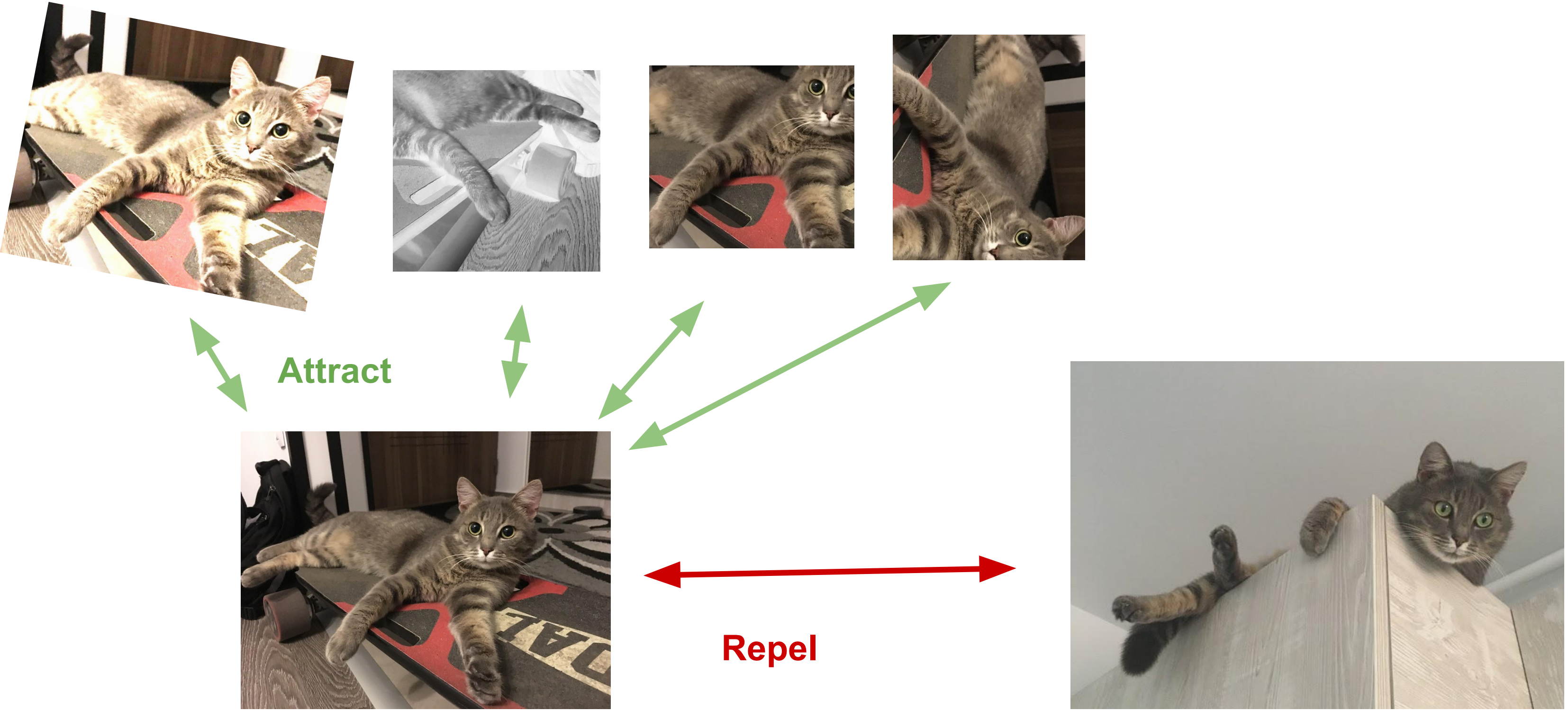


Problem: A model outputting a vector of zeros satisfies this condition

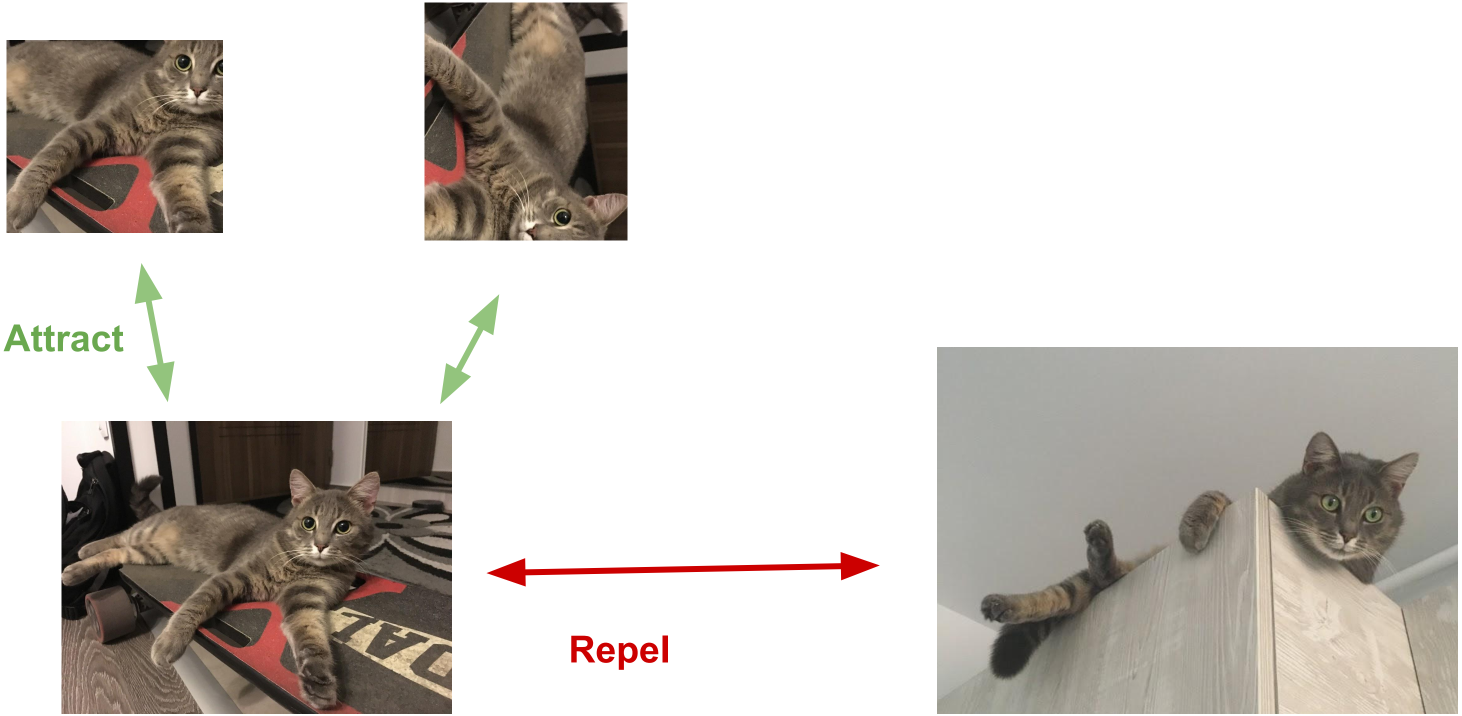
(embedding collapse)

Contrastive Learning

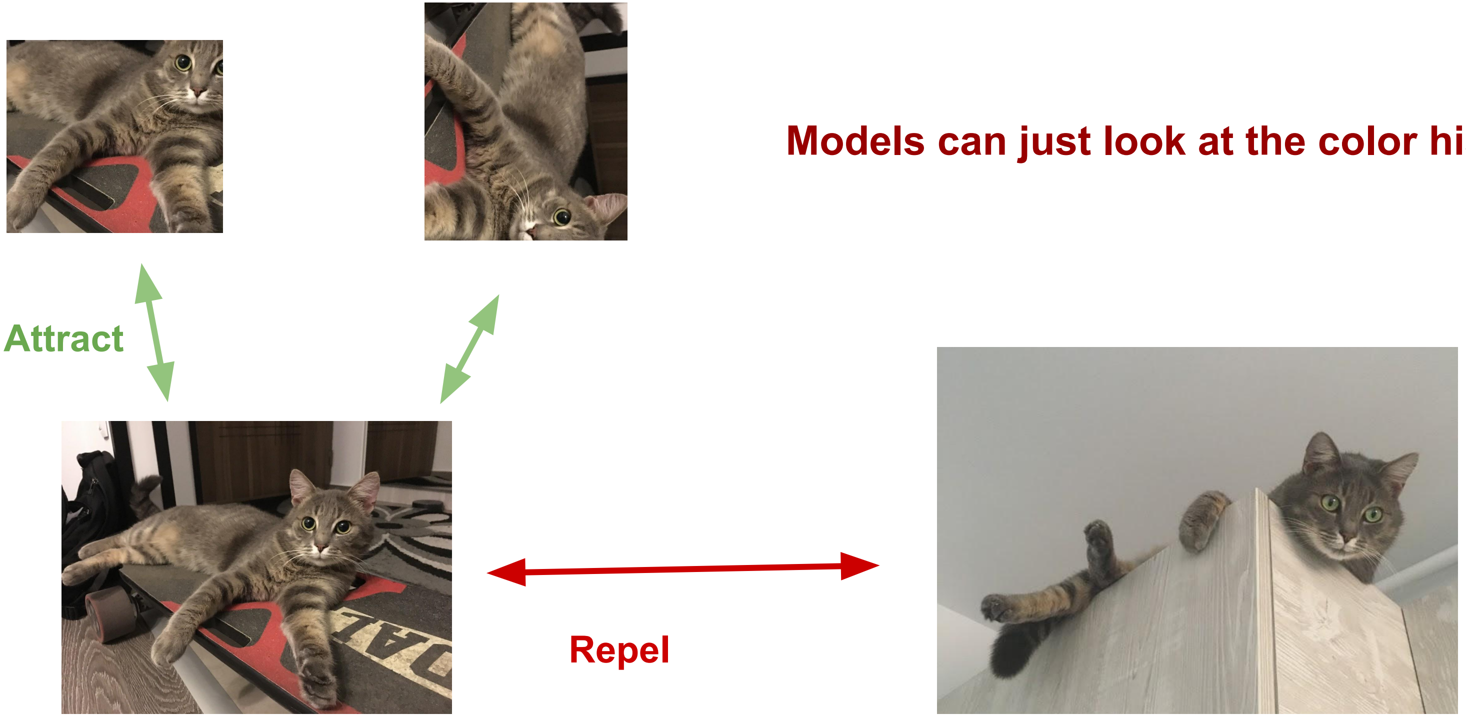
Solution: Attract representations from the same image, and repel representation from a different image



How can models cheat in this task?



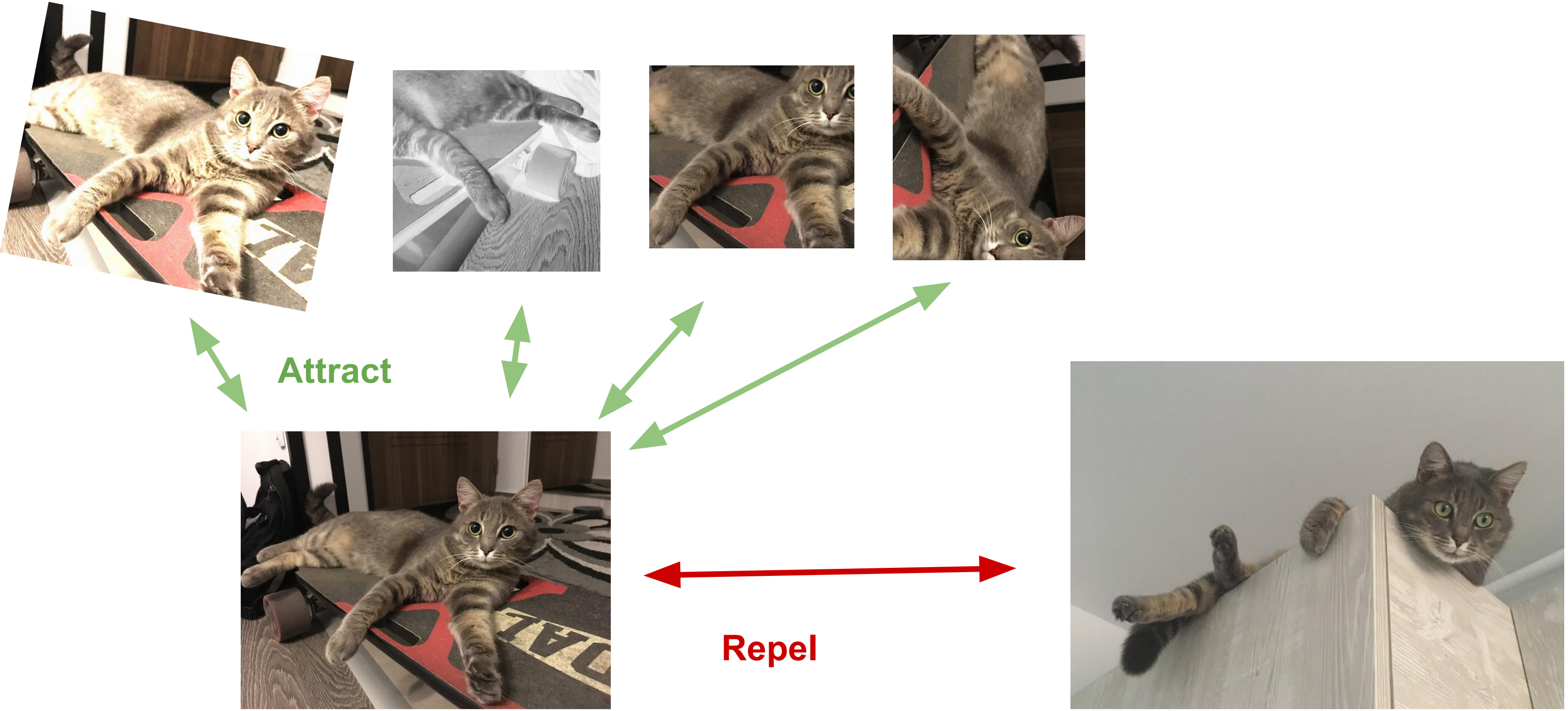
How can models cheat in this task?



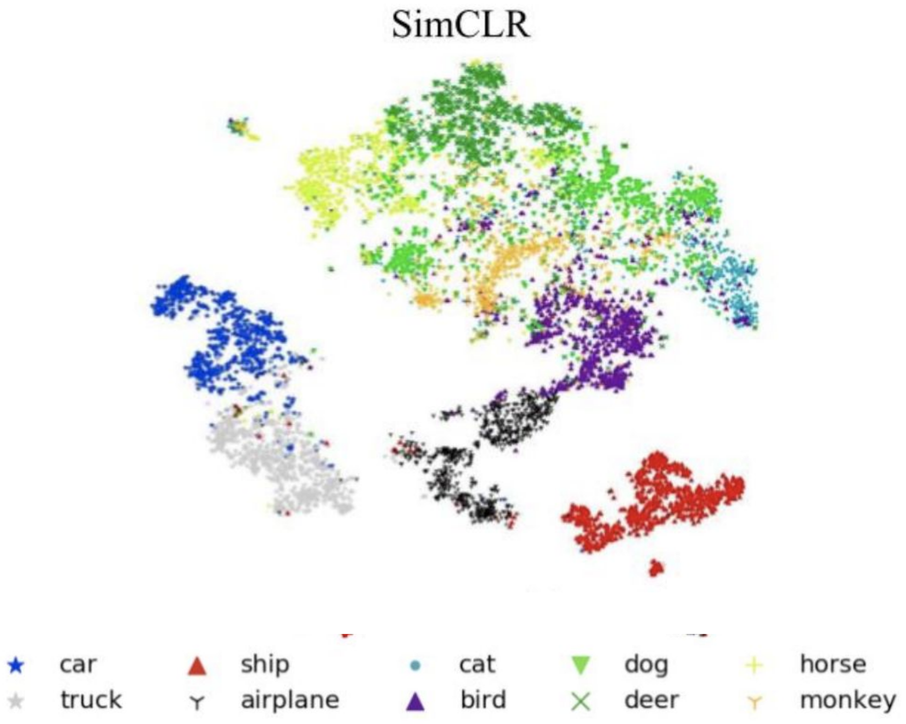
Models can just look at the color histogram!

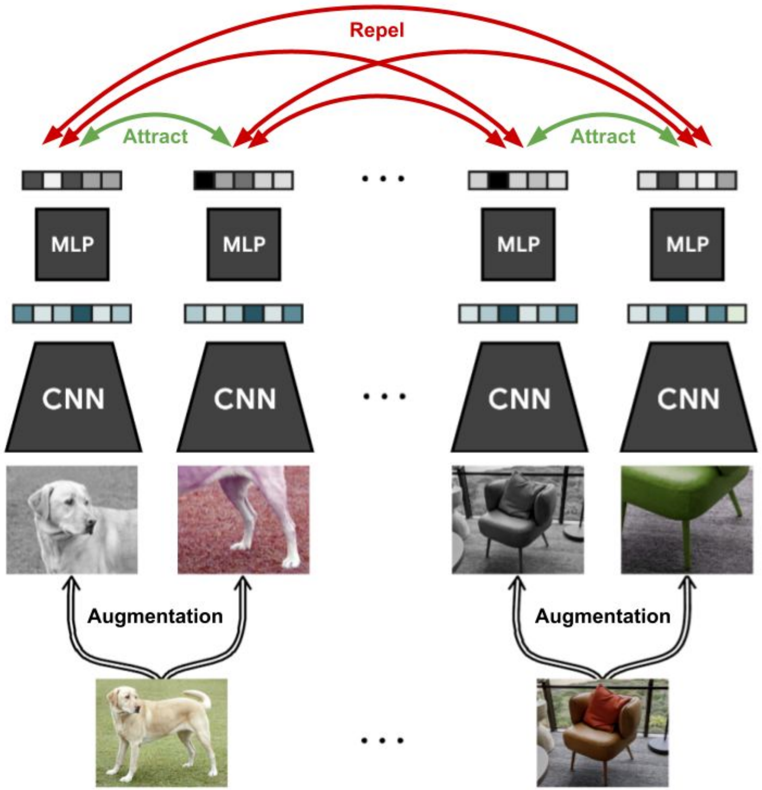
How can models cheat in this task?

Important! Use **ColorJitter!**



SimCLR: A Simple Framework for Contrastive Learning





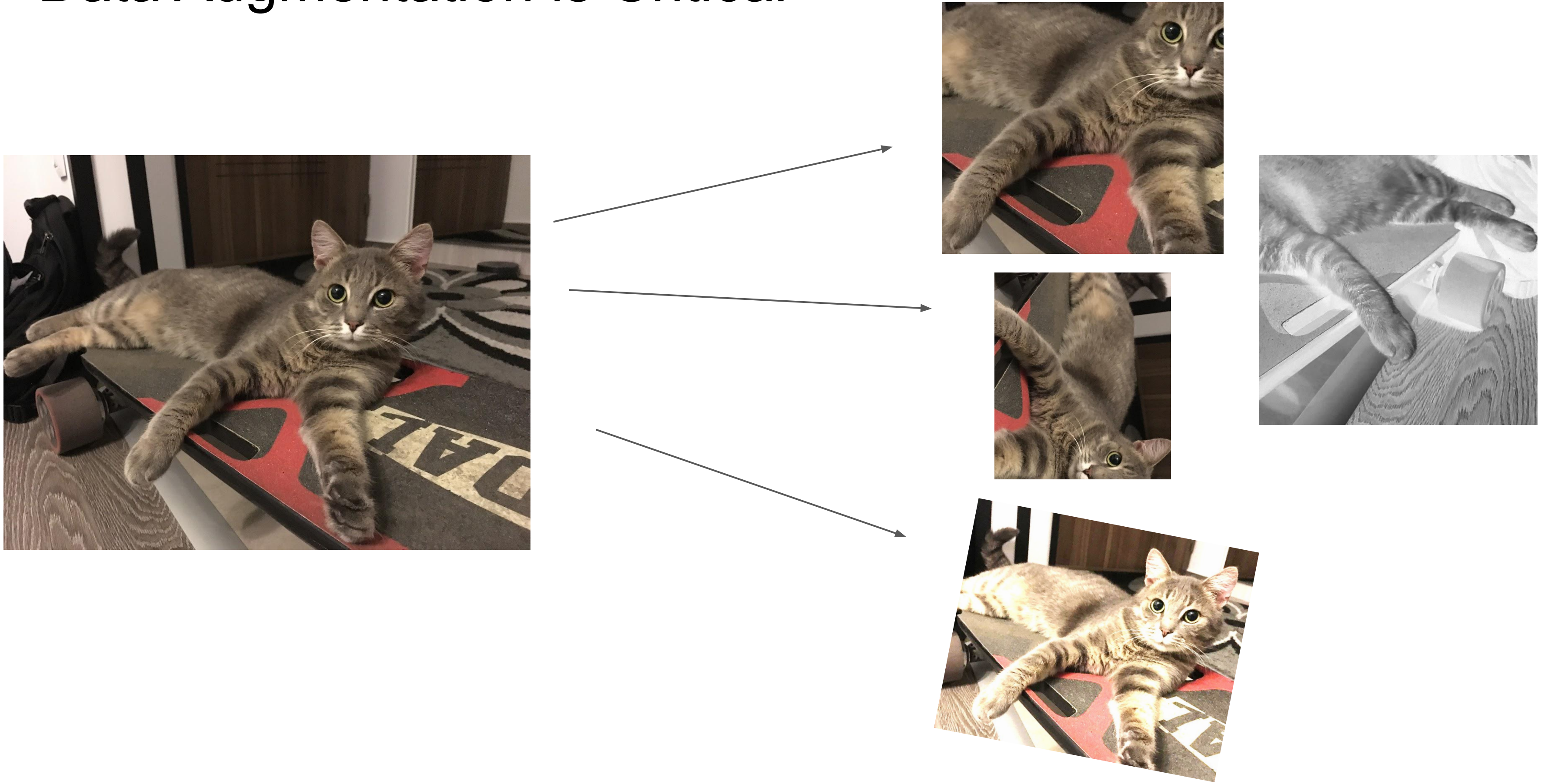
Data Augmentation is Critical

Programmatically generate new variations of an input

- Increase data variation
- Cheap + fast

Requirement: Must not change the underlying class (an augmented cat must remain a cat)

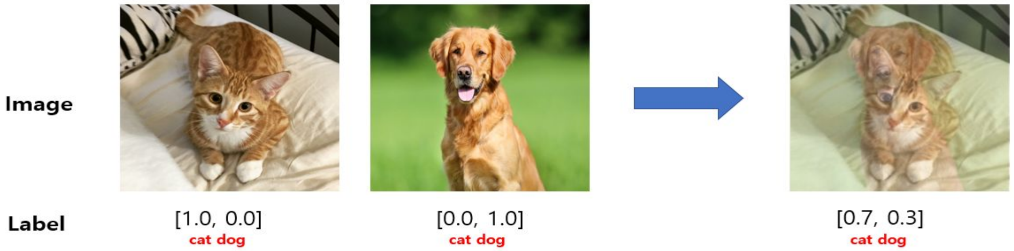
Data Augmentation is Critical



Data Augmentation is Critical

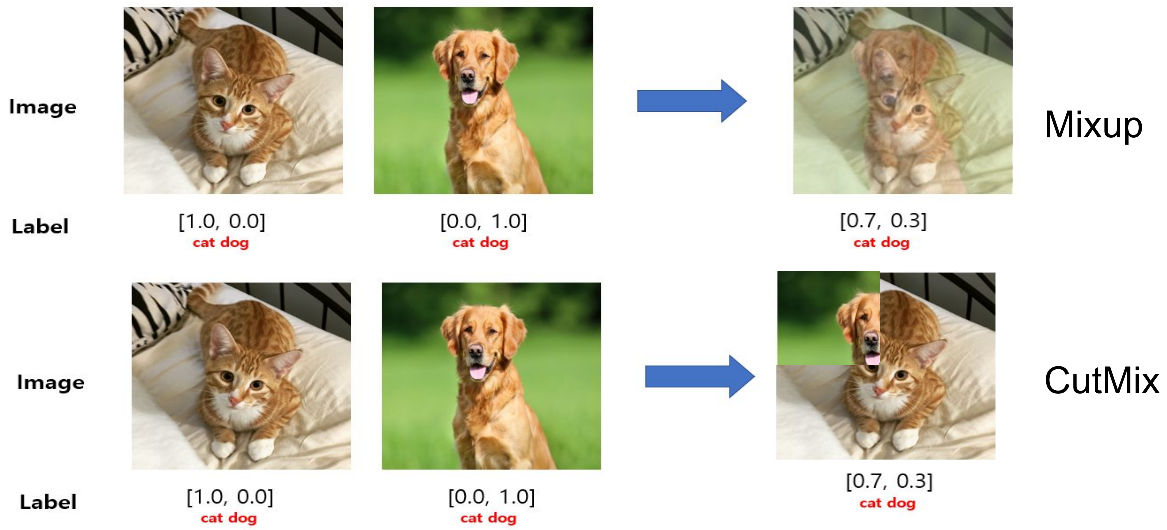
- Common data augmentations
  - Random Crops
  - Horizontal Flips
  - Vertical Flips
  - Rotations
  - Color Jitter
  - Brightness / Contrast
  - Random Blur
  - Cutout
  - etc

Other Augmentations - Image Mixup / CutMix

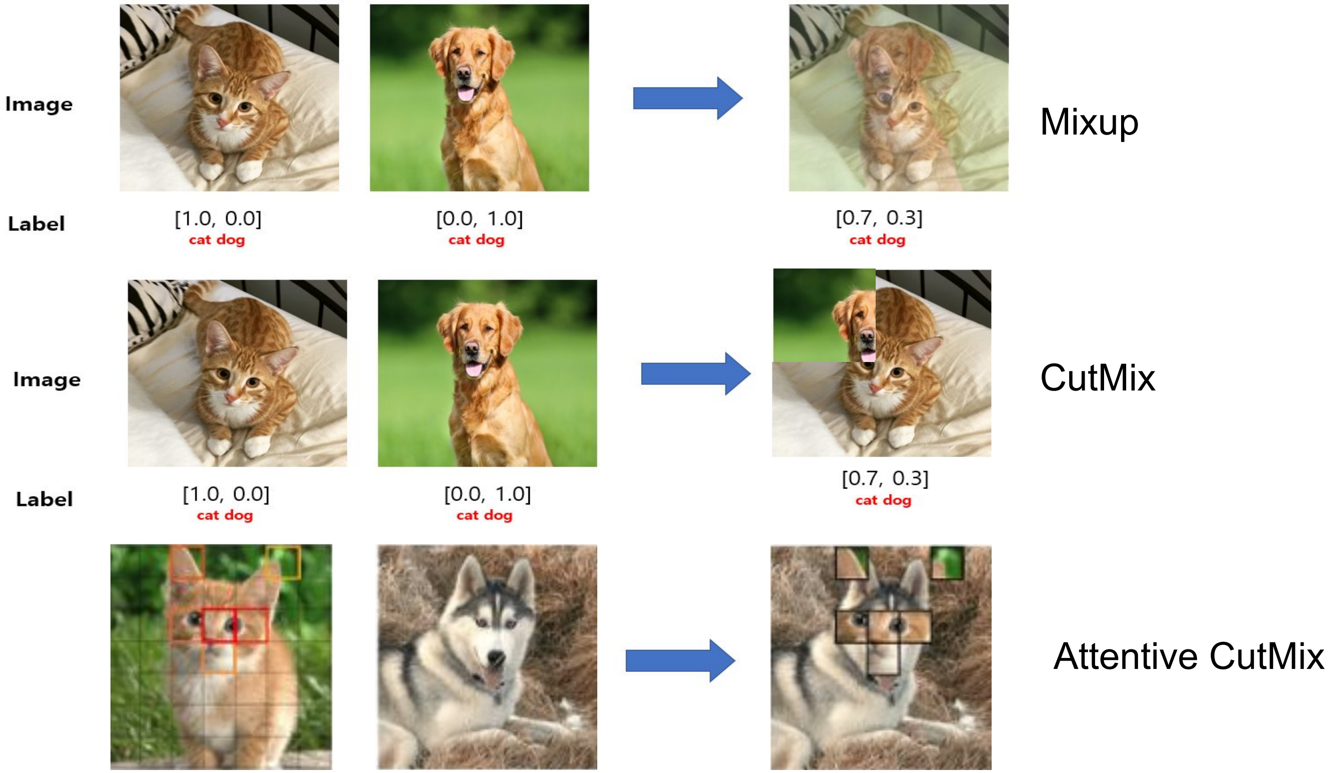


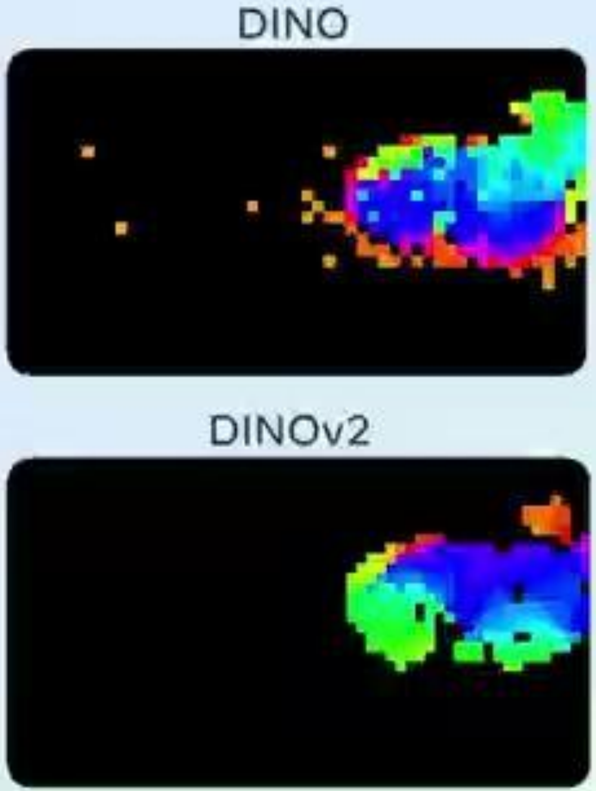
Mixup

Other Augmentations - Image Mixup / CutMix



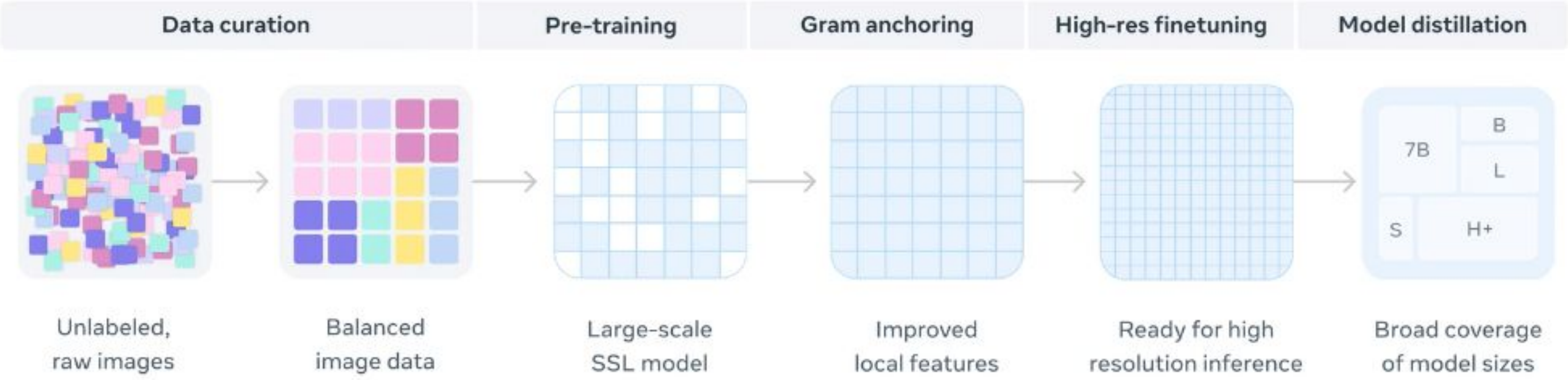
Other Augmentations - Image Mixup / CutMix




DINO
DINOv2

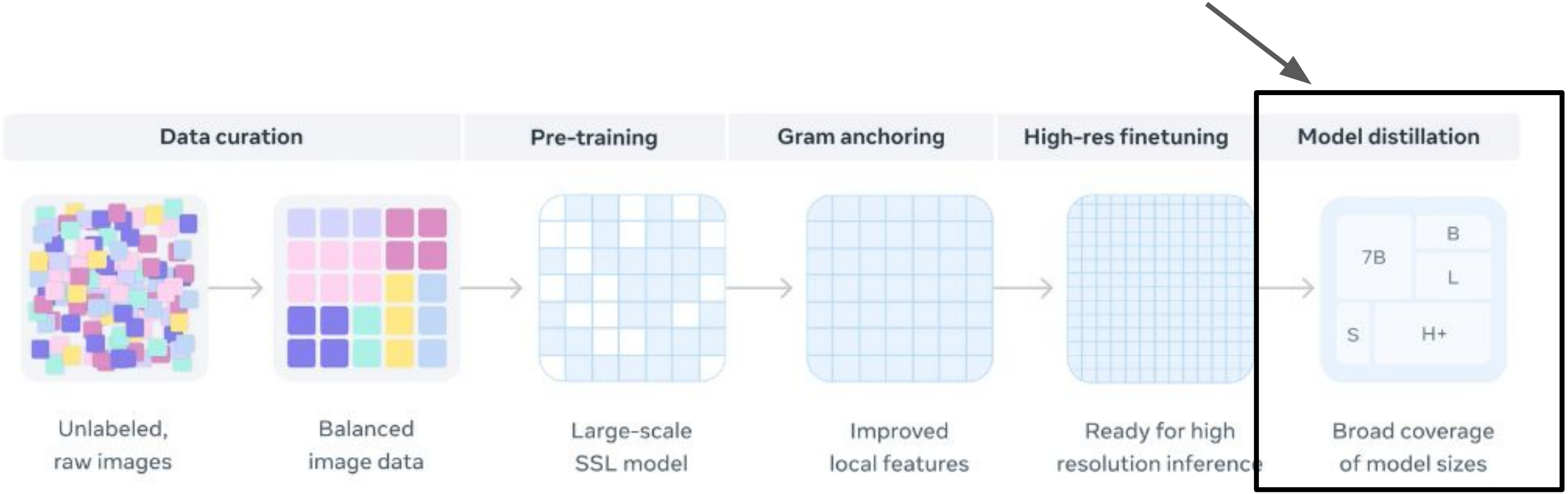
DINOv3 Training Pipeline

- Similar pattern:
  - Train big model on large-scale data
  - Distill knowledge into smaller models



DINOv3 Training Pipeline

- Similar pattern:
  - Train big model on large-scale data
  - Distill knowledge into smaller models



Observation: Unsupervised Learning is ill-posed

- Old dream of ML
- Does not really show up at small scale

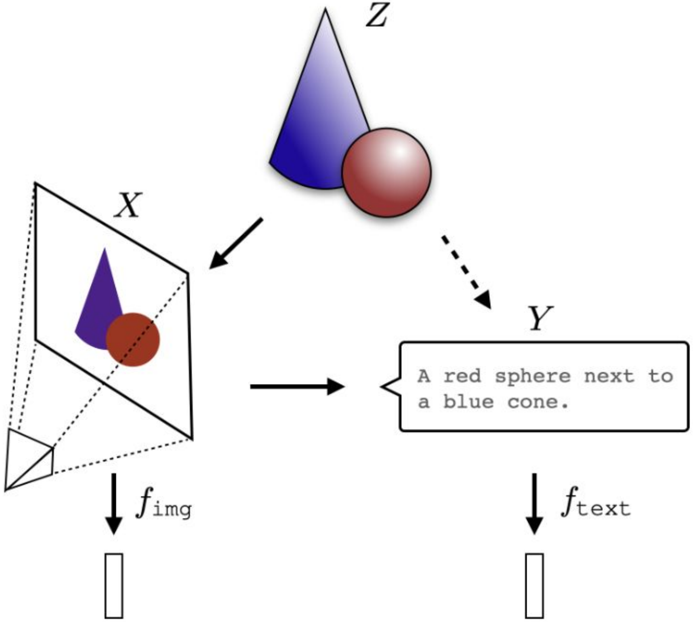
- Optimize for one objective (i.e., contrastive loss)
  - But we care about another objective! (e.g., classification accuracy)

The Platonic Representation Hypothesis

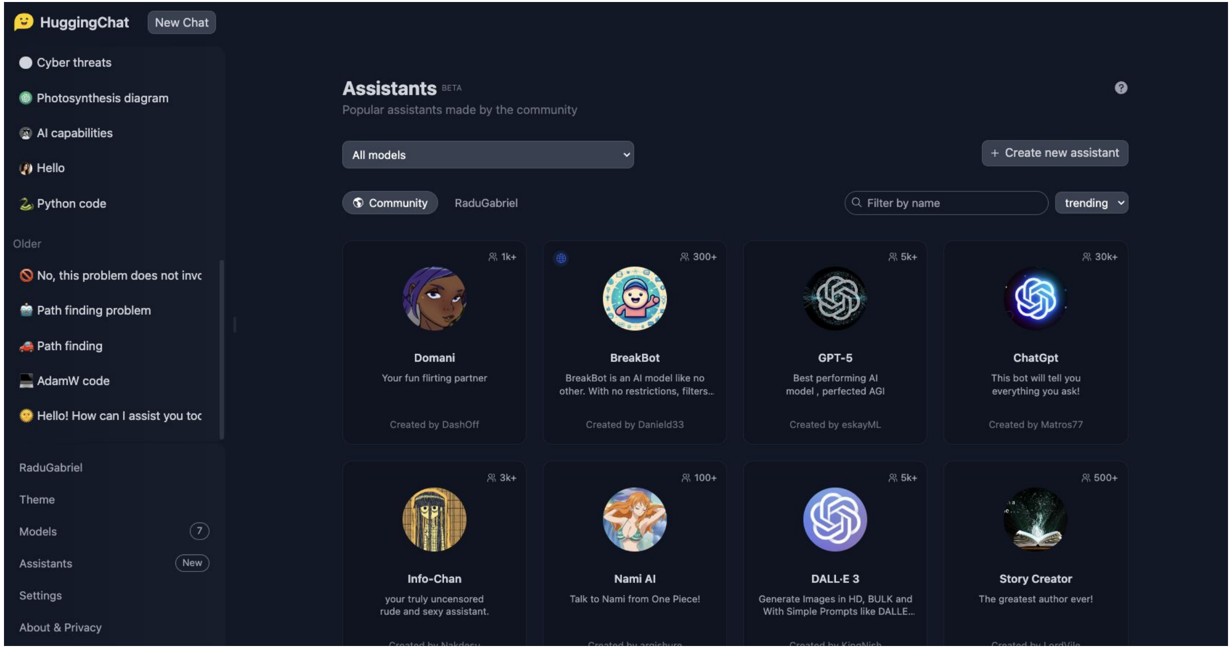
- With enough scale / data, models capture a common representation of reality, irrespective of objective

- Intuitively, because models **compress** the data in a similar way
    - No time for this now; it's a rabbit hole

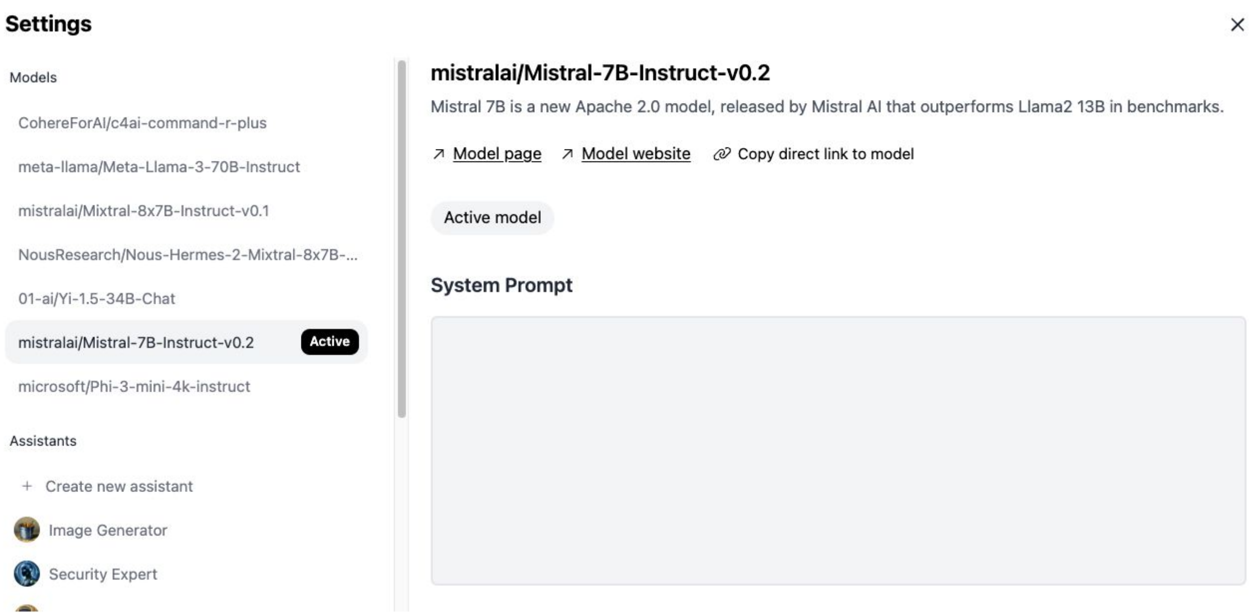
> **The Platonic Representation Hypothesis**
>
> Neural networks, trained with different objectives on different data and modalities, are converging to a shared statistical model of reality in their representation spaces.

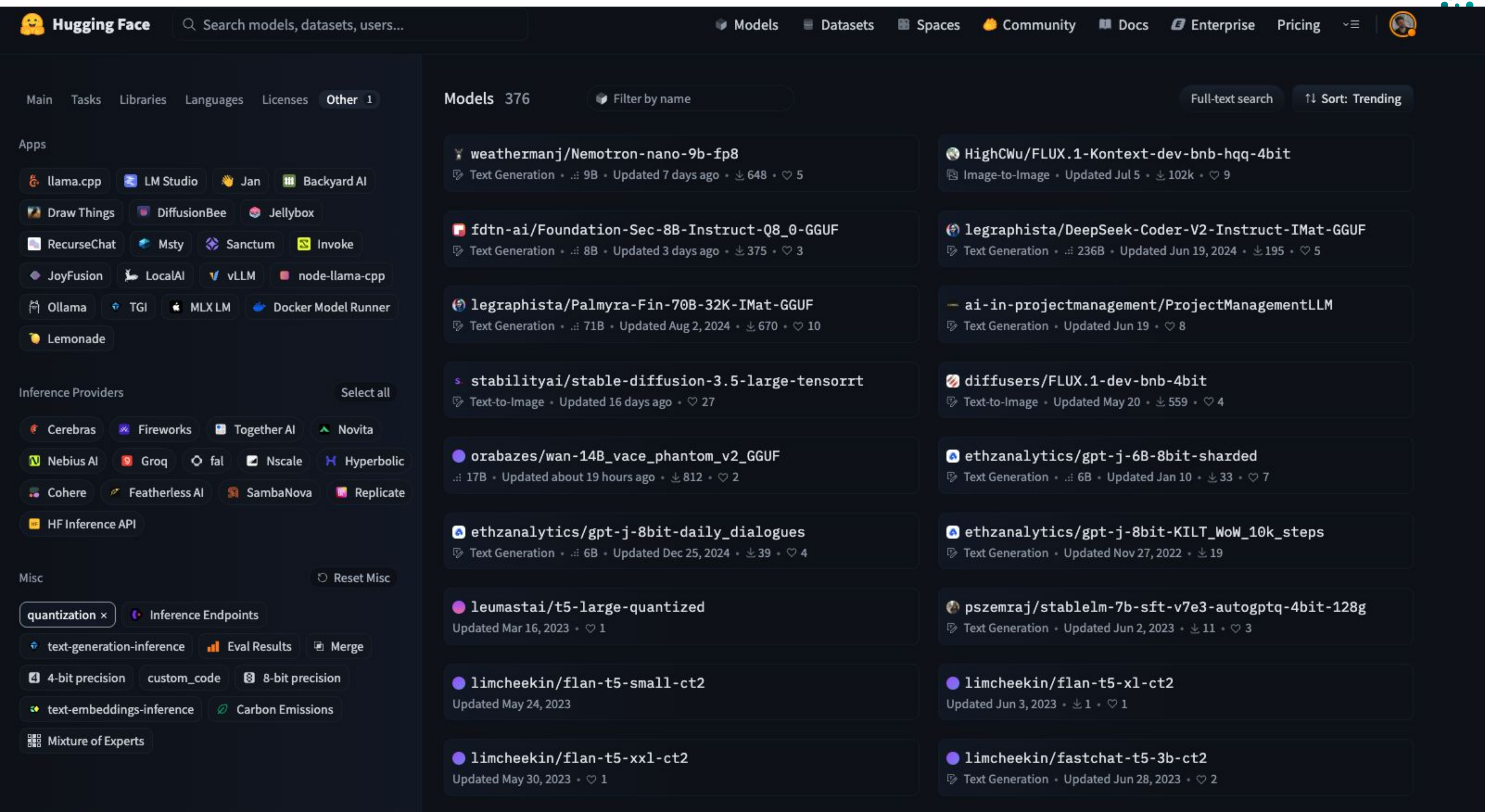


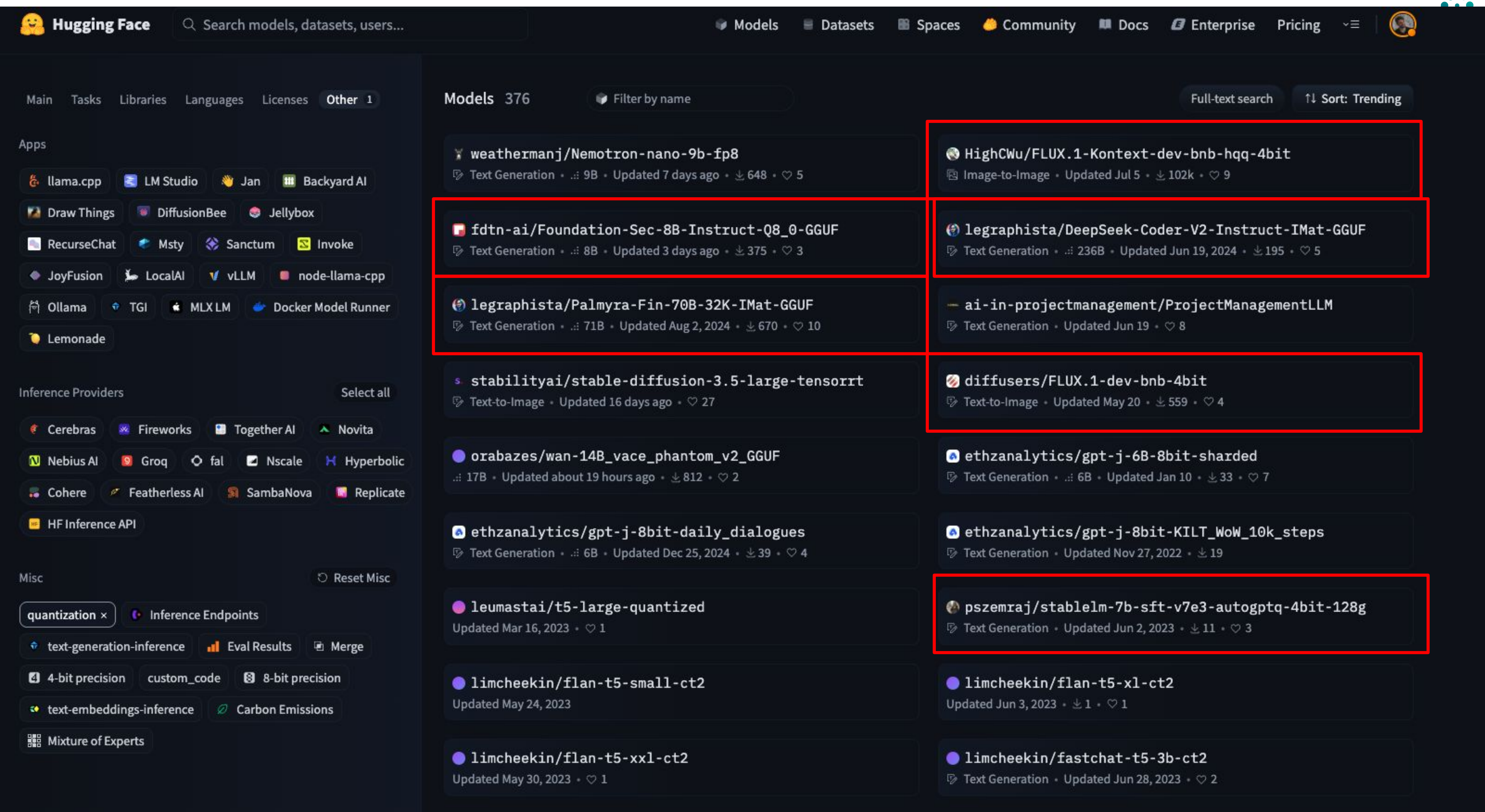
Huggingface - The Github for AI Models / Datasets



Huggingface - The Github for AI Models / Datasets




Hugging Face
Search models, datasets, users...
Models
Datasets
Spaces
Community
Docs
Enterprise
Pricing
Main
Tasks
Libraries
Languages
Licenses
Other 1
Apps
llama.cpp
LM Studio
Jan
Backyard AI
Draw Things
DiffusionBee
Jellybox
RecurseChat
Msty
Sanctum
Invoke
JoyFusion
LocalAI
vLLM
node-llama-cpp
Ollama
TGI
MLX LM
Docker Model Runner
Lemonade
Inference Providers
Select all
Cerebras
Fireworks
Together AI
Novita
Nebius AI
Groq
fal
Nscale
Hyperbolic
Cohere
Featherless AI
SambaNova
Replicate
HF Inference API
Misc
Reset Misc
quantization ×
Inference Endpoints
text-generation-inference
Eval Results
Merge
4-bit precision
custom_code
8-bit precision
text-embeddings-inference
Carbon Emissions
Mixture of Experts
Models 376
Filter by name
Full-text search
Sort: Trending
weathermanj/Nemotron-nano-9b-fp8
Text Generation • 9B • Updated 7 days ago • 648 • 5
HighCWu/FLUX.1-Kontext-dev-bnb-hqq-4bit
Image-to-Image • Updated Jul 5 • 102k • 9
fdtn-ai/Foundation-Sec-8B-Instruct-Q8_0-GGUF
Text Generation • 8B • Updated 3 days ago • 375 • 3
legraphista/DeepSeek-Coder-V2-Instruct-IMat-GGUF
Text Generation • 236B • Updated Jun 19, 2024 • 195 • 5
legraphista/Palmyra-Fin-70B-32K-IMat-GGUF
Text Generation • 71B • Updated Aug 2, 2024 • 670 • 10
ai-in-projectmanagement/ProjectManagementLLM
Text Generation • Updated Jun 19 • 8
stabilityai/stable-diffusion-3.5-large-tensorrt
Text-to-Image • Updated 16 days ago • 27
diffusers/FLUX.1-dev-bnb-4bit
Text-to-Image • Updated May 20 • 559 • 4
orabazes/wan-14B_vace_phantom_v2_GGUF
17B • Updated about 19 hours ago • 812 • 2
ethzanalytics/gpt-j-6B-8bit-sharded
Text Generation • 6B • Updated Jan 10 • 33 • 7
ethzanalytics/gpt-j-8bit-daily_dialogues
Text Generation • 6B • Updated Dec 25, 2024 • 39 • 4
ethzanalytics/gpt-j-8bit-KILT_WoW_10k_steps
Text Generation • Updated Nov 27, 2022 • 19
leumastai/t5-large-quantized
Updated Mar 16, 2023 • 1
pszemraj/stablelm-7b-sft-v7e3-autogptq-4bit-128g
Text Generation • Updated Jun 2, 2023 • 11 • 3
limcheekin/flan-t5-small-ct2
Updated May 24, 2023
limcheekin/flan-t5-xl-ct2
Updated Jun 3, 2023 • 1 • 1
limcheekin/flan-t5-xxl-ct2
Updated May 30, 2023 • 1
limcheekin/fastchat-t5-3b-ct2
Text Generation • Updated Jun 28, 2023 • 2


Hugging Face
Search models, datasets, users...
Models
Datasets
Spaces
Community
Docs
Enterprise
Pricing
Main
Tasks
Libraries
Languages
Licenses
Other 1
Apps
llama.cpp
LM Studio
Jan
Backyard AI
Draw Things
DiffusionBee
Jellybox
RecurseChat
Msty
Sanctum
Invoke
JoyFusion
LocalAI
vLLM
node-llama-cpp
Ollama
TGI
MLX LM
Docker Model Runner
Lemonade
Inference Providers
Select all
Cerebras
Fireworks
Together AI
Novita
Nebius AI
Groq
fal
Nscale
Hyperbolic
Cohere
Featherless AI
SambaNova
Replicate
HF Inference API
Misc
Reset Misc
quantization ×
Inference Endpoints
text-generation-inference
Eval Results
Merge
4-bit precision
custom_code
8-bit precision
text-embeddings-inference
Carbon Emissions
Mixture of Experts
Models 376
Filter by name
Full-text search
Sort: Trending
weathermanj/Nemotron-nano-9b-fp8
Text Generation • 9B • Updated 7 days ago • 648 • 5
HighCWu/FLUX.1-Kontext-dev-bnb-hqq-4bit
Image-to-Image • Updated Jul 5 • 102k • 9
fdtn-ai/Foundation-Sec-8B-Instruct-Q8_0-GGUF
Text Generation • 8B • Updated 3 days ago • 375 • 3
legraphista/DeepSeek-Coder-V2-Instruct-IMat-GGUF
Text Generation • 236B • Updated Jun 19, 2024 • 195 • 5
legraphista/Palmyra-Fin-70B-32K-IMat-GGUF
Text Generation • 71B • Updated Aug 2, 2024 • 670 • 10
ai-in-projectmanagement/ProjectManagementLLM
Text Generation • Updated Jun 19 • 8
stabilityai/stable-diffusion-3.5-large-tensorrt
Text-to-Image • Updated 16 days ago • 27
diffusers/FLUX.1-dev-bnb-4bit
Text-to-Image • Updated May 20 • 559 • 4
orabazes/wan-14B_vace_phantom_v2_GGUF
17B • Updated about 19 hours ago • 812 • 2
ethzanalytics/gpt-j-6B-8bit-sharded
Text Generation • 6B • Updated Jan 10 • 33 • 7
ethzanalytics/gpt-j-8bit-daily_dialogues
Text Generation • 6B • Updated Dec 25, 2024 • 39 • 4
ethzanalytics/gpt-j-8bit-KILT_WoW_10k_steps
Text Generation • Updated Nov 27, 2022 • 19
leumastai/t5-large-quantized
Updated Mar 16, 2023 • 1
pszemraj/stablelm-7b-sft-v7e3-autogptq-4bit-128g
Text Generation • Updated Jun 2, 2023 • 11 • 3
limcheekin/flan-t5-small-ct2
Updated May 24, 2023
limcheekin/flan-t5-xl-ct2
Updated Jun 3, 2023 • 1 • 1
limcheekin/flan-t5-xxl-ct2
Updated May 30, 2023 • 1
limcheekin/fastchat-t5-3b-ct2
Text Generation • Updated Jun 28, 2023 • 2

Tasks in Computer Vision

- Scene text reading (OCR)
- Object recognition (classification)
- Object delineation (detection, segmentation)
- Chart / infographic parsing
- Document parsing
- Instrument reading
- Place recognition
- Action recognition
- Face recognition
- World knowledge
- Visual question answering

Tasks in Computer Vision

- ~~Scene text reading (OCR)~~
- ~~Object recognition (classification)~~
- ~~Object delineation (detection, segmentation)~~
- ~~Chart / infographic parsing~~
- ~~Document parsing~~
- ~~Instrument reading~~
- ~~Place recognition~~
- ~~Action recognition~~
- ~~Face recognition~~
- ~~World knowledge~~
- **Visual question answering**

Fake Tasks

Real Tasks

Workshop Overview

1. Part 0: Introduction to the Team & Rust for Edge AI
2. **Part I: Lecture on Computer Vision**
    a. Main problems in Computer Vision
    b. What exactly is a neural network? (CNNs / Transformers)
    c. What exactly is an image embedding?
    d. Computer vision on the Edge
3. Hands-On I: Air-gapped Face recognition on the Pi
4. Part II: Lecture on Natural Language Processing
    a. A bit of history & development of modern LLMs
    b. How does an LLM work? Tokenizers, pretraining, post-training
    c. Context Engineering: Tool calling, RAG
    d. Libraries: tokenizers-rs, llama.cpp
5. Hands-On II: Chat with a LLM on Pi
6. Hands-On III: Knight Rider

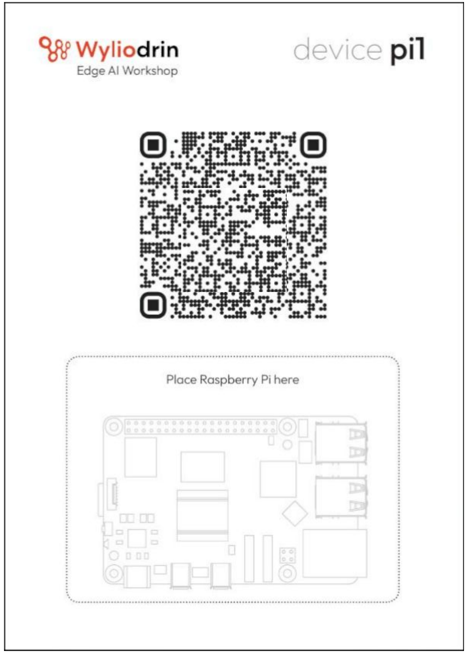
Workshop Overview

1. **Part 0: Introduction to the Team & Rust for Edge AI**
2. **Part I: Lecture on Computer Vision**
    a. Main problems in Computer Vision
    b. What exactly is a neural network? (CNNs / Transformers)
    c. What exactly is an image embedding?
    d. Computer vision on the Edge
3. **Hands-On I: Air-gapped Face recognition on the Pi**
4. **Part II: Lecture on Natural Language Processing**
    a. A bit of history & development of modern LLMs
    b. How does an LLM work? Tokenizers, pretraining, post-training
    c. Context Engineering: Tool calling, RAG
    d. Libraries: tokenizers-rs, llama.cpp
5. **Hands-On II: Chat with a LLM on Pi**
6. **Hands-On III: Knight Rider**

Connections

Laptop

WiFi: RustConf2025_AI / `edgeaiworkshop`



Raspberry Pi

Start the Pi (plug the SD Card and the power adapter)

Scan the QR Code for the Raspberry Pi’s IP address

Login: `pi` / `edgeaiworkshop`

`Use VSCode or Zed (Linux or macOS) Remote Connection`

Hands-on Overview
Privacy-Preserving, Local-Device, Facial Recognition

- Use candle to instantiate and run inference for a ConvNext model
- Compute image embeddings and implement a basic vector storage
- Integrate in a POC application for real time user register and login

https://github.com/Wyliodrin/edge-ai-face-auth