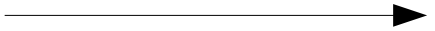
Deep Learning

Episode 3 fall'22

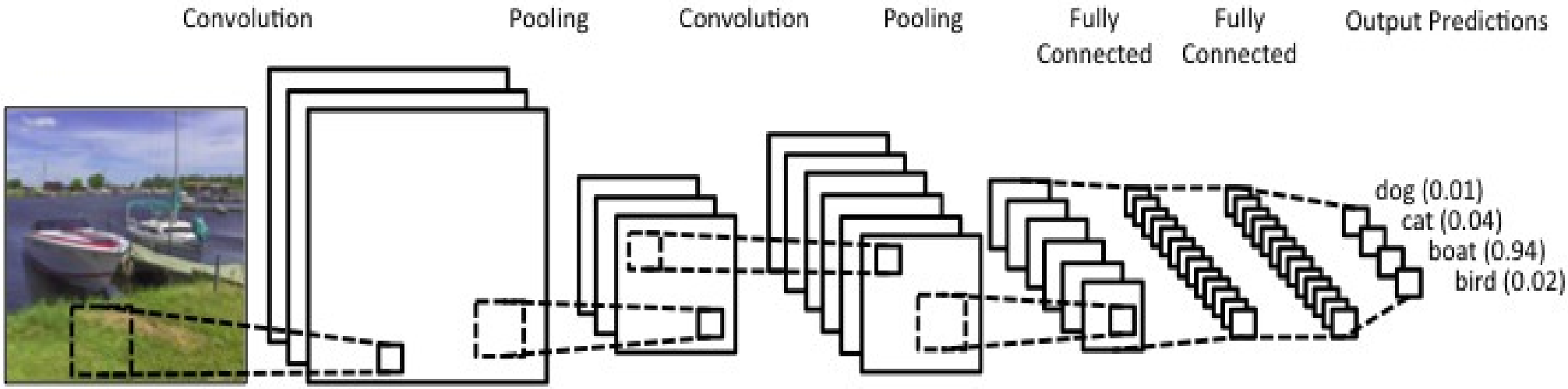
Transfer learning & more vision tasks

Previously...

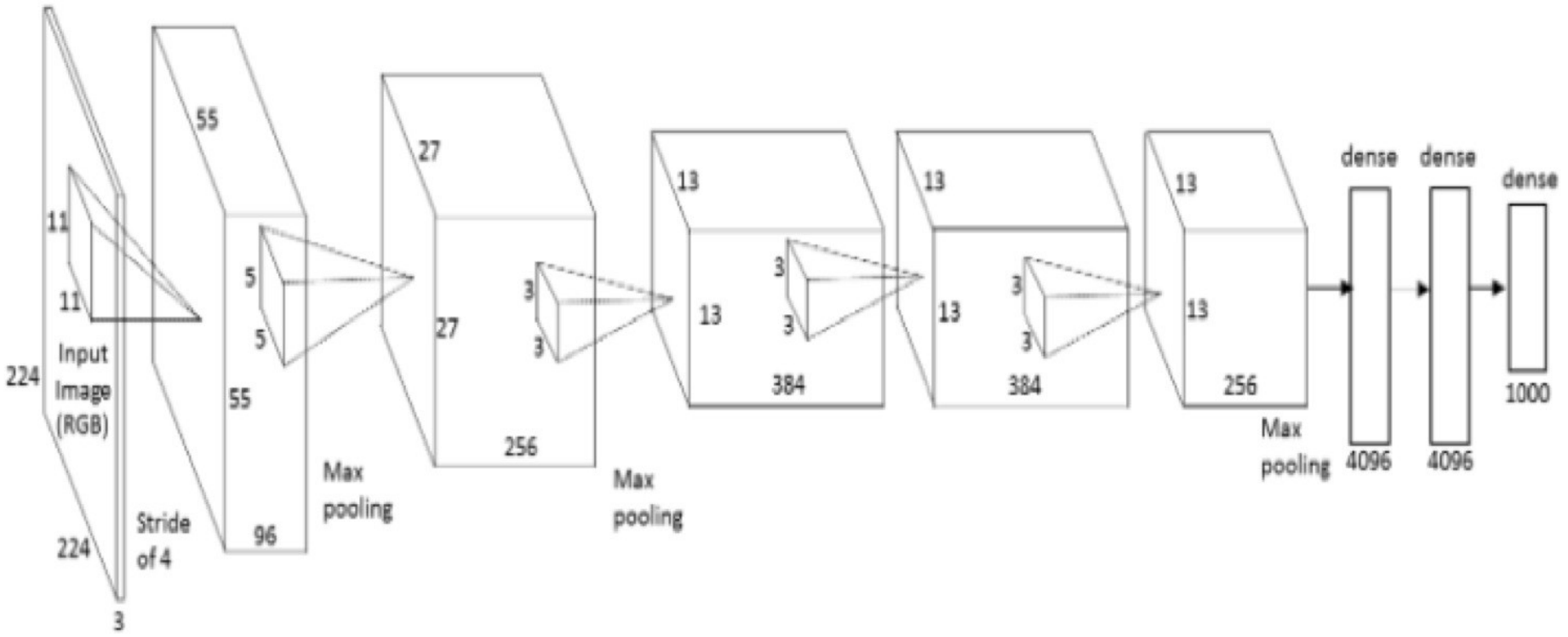


“Dog”

Previously...

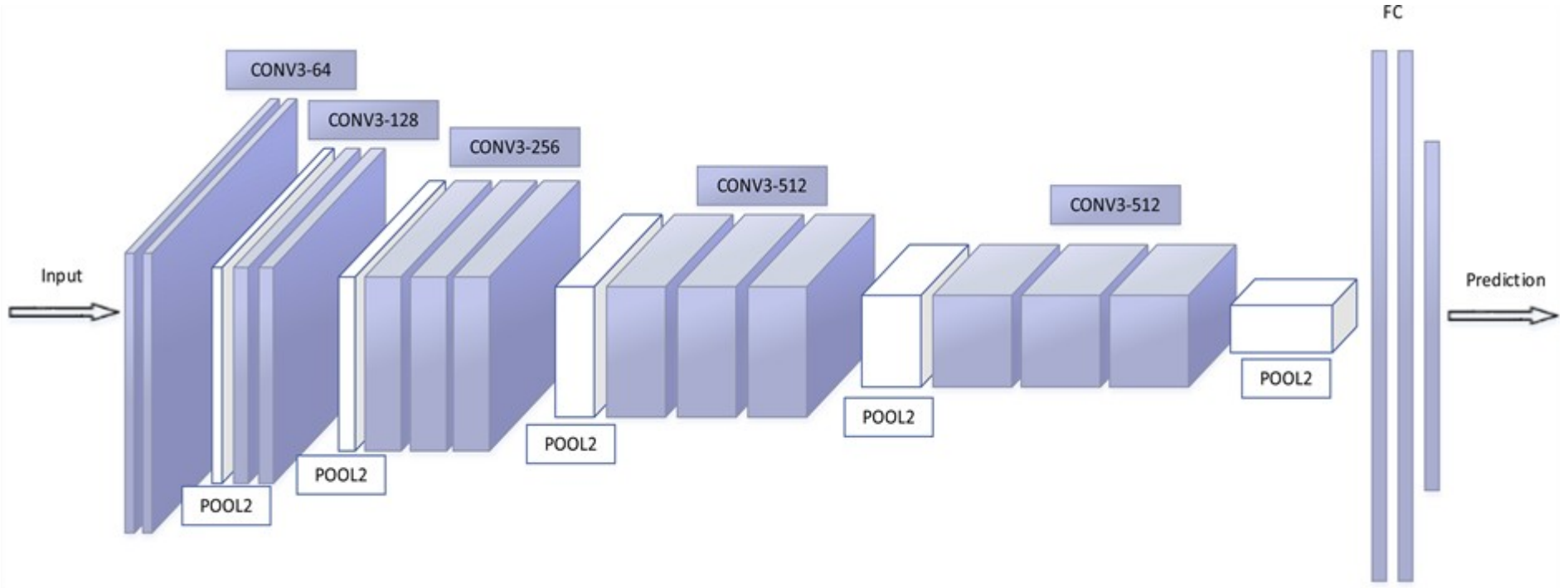

Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

Alexnet



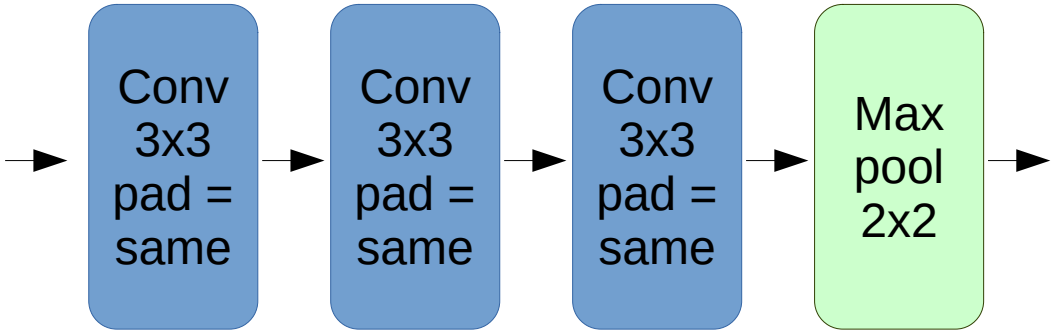
(Krizhevsky et al.)

VGG16

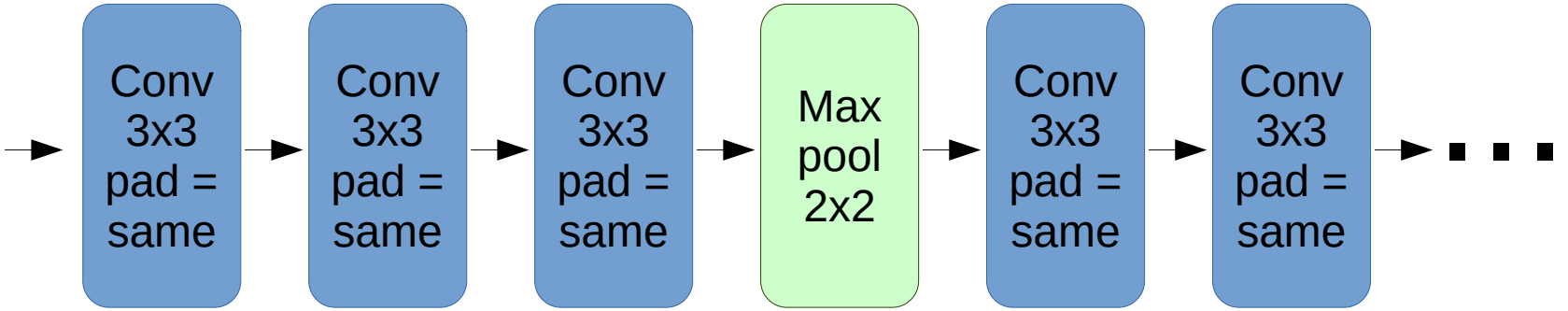


Karen Simonyan and Andrew Zisserman

Previously...


Conv 3x3 pad = same
Conv 3x3 pad = same
Conv 3x3 pad = same
Max pool 2x2

Previously...


Conv 3x3 pad = same
Conv 3x3 pad = same
Conv 3x3 pad = same
Max pool 2x2
Conv 3x3 pad = same
Conv 3x3 pad = same

Previously...

Conv 3x3 pad = same
Conv 3x3 pad = same
Conv 3x3 pad = same
Max pool 2x2
Conv 3x3 pad = same
Conv 3x3 pad = same
Conv 3x3 pad = same
Conv 3x3 pad = same
Max pool 2x2
Conv 3x3 pad = same
Conv 3x3 pad = same
Max pool 2x2
Conv 3x3 pad = same

Previously...

Conv 3x3 pad = same → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → Conv 3x3 pad = same → ...

... → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → ...

... → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2

???

How to predict class?

Previously...

Conv 3x3 pad = same → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → Conv 3x3 pad = same → . . .

. . . → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → . . .

. . . → Conv 3x3 pad = same → Max pool 2x2 → Conv 3x3 pad = same → Conv 3x3 pad = same → Max pool 2x2 → Dense 1000 → Dense 1000 softmax → “dog”

Previously...

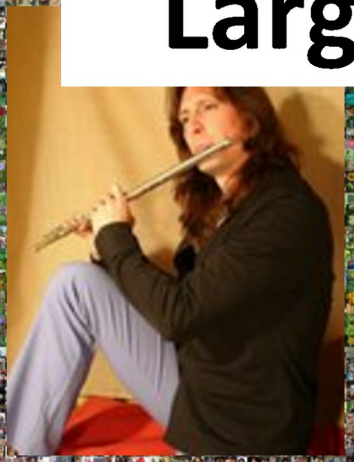
What you sign for if you stack 1000 layers:

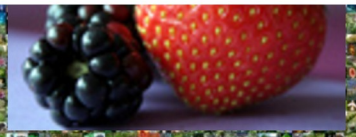
- MemoryError(0x...)
- Activations can vanish
- Activations can explode
- Gradients can vanish
- Gradients can explode

Batch normalization
or similar

???

Large-scale recognition





CIFAR

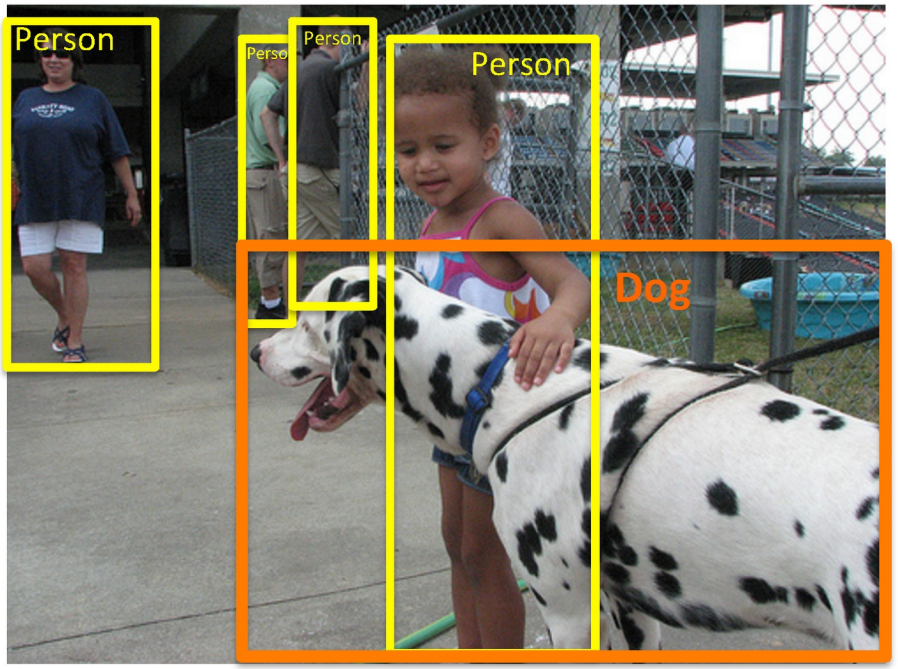
- 60k images
- 10 classes (left)
- 32x32 RGB
- subclasses

aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates

Pfft... weakling

IMAGENET Large Scale Visual Recognition Challenge (ILSVRC) 2010-2014

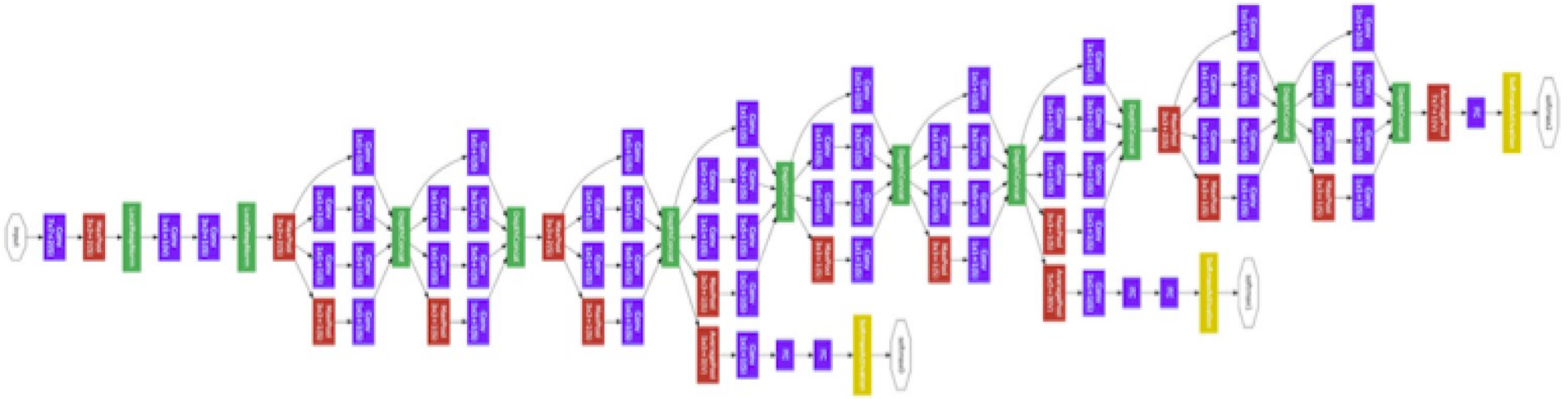
200 object classes	**456,567 images**	**DET**
1000 object classes	**1,431,167 images**	**CLS-LOC**



http://image-net.org/challenges/LSVRC/

Variety of object classes in ILSVRC

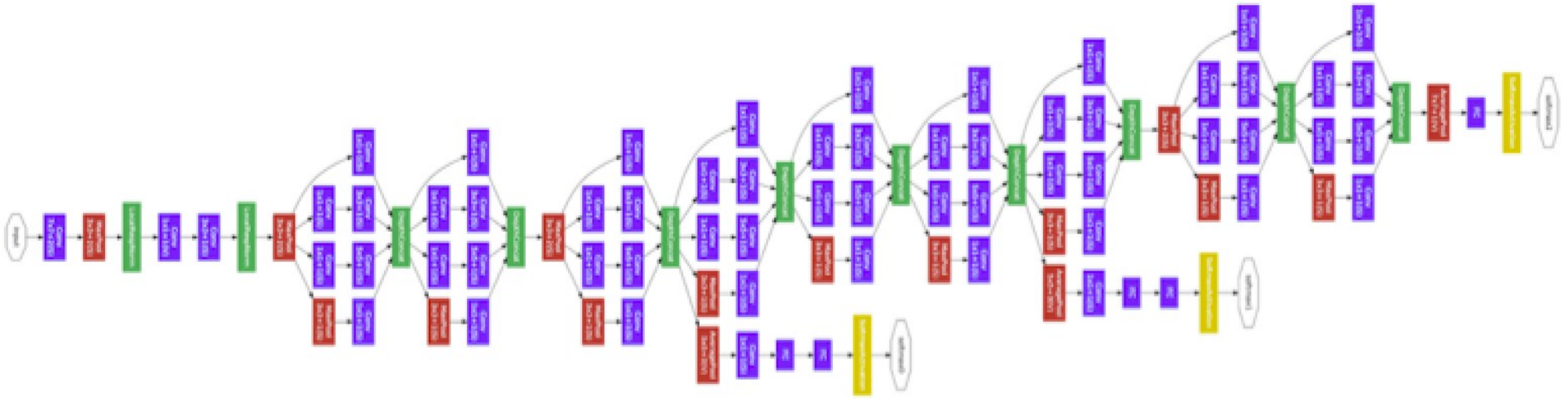
Recap: Inception



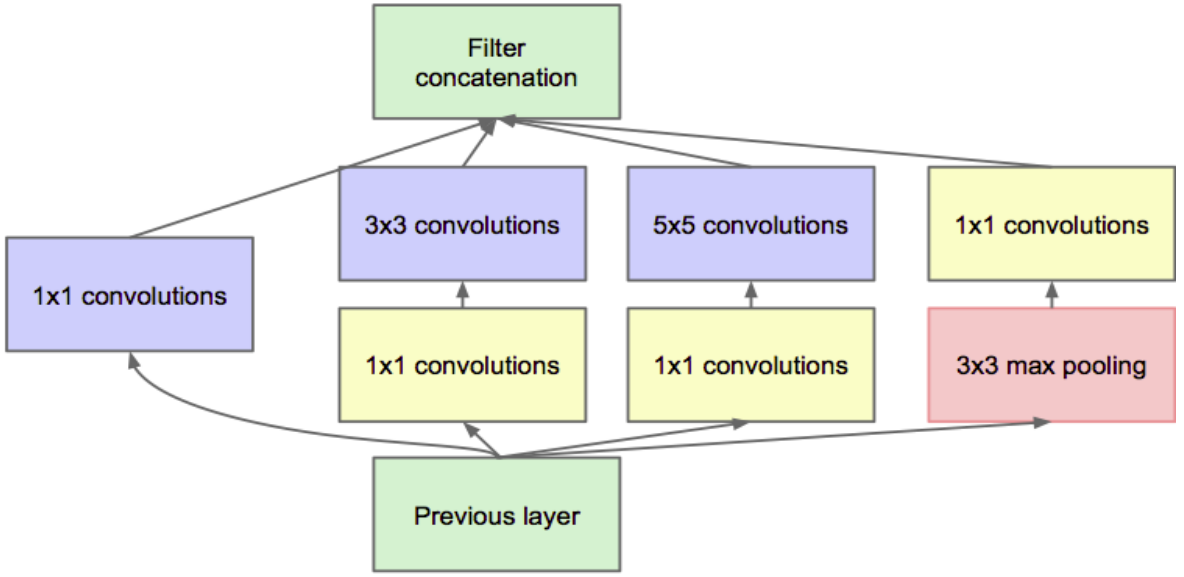
Convolution
Pooling
Softmax
Other

It is not a moon. It is a space station (c)

Recap: Inception



"Inception module"



Convolution
Pooling
Softmax
Other

Recap: Inception

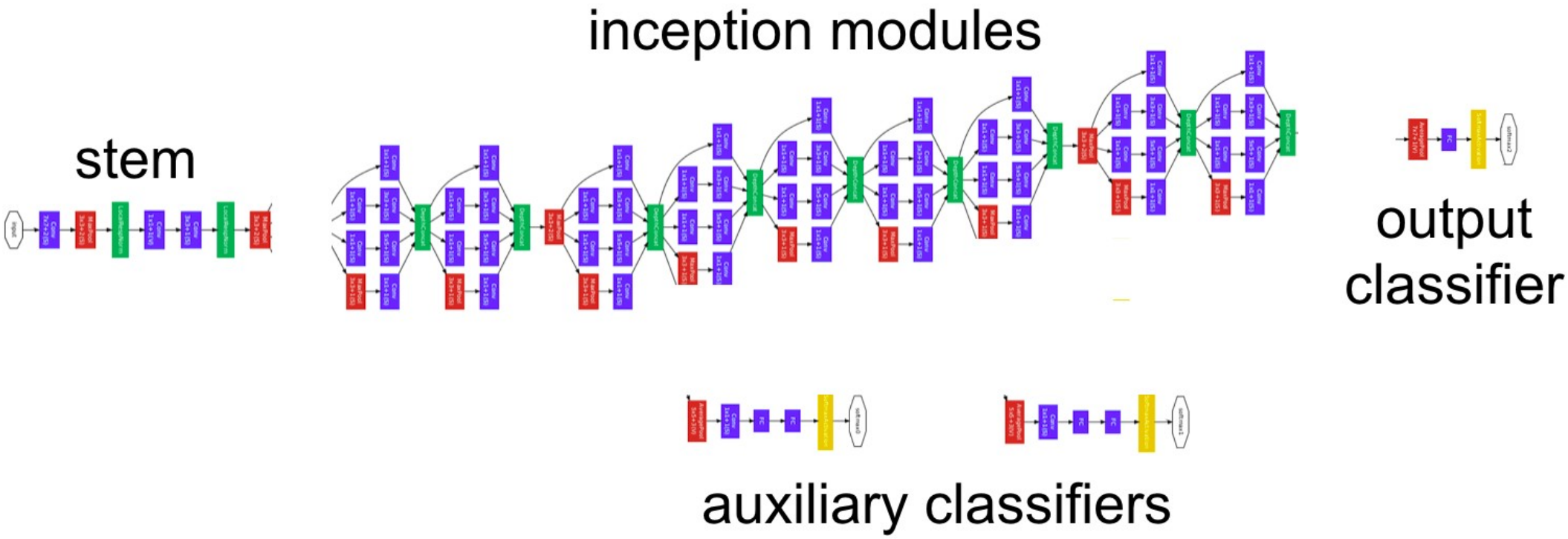
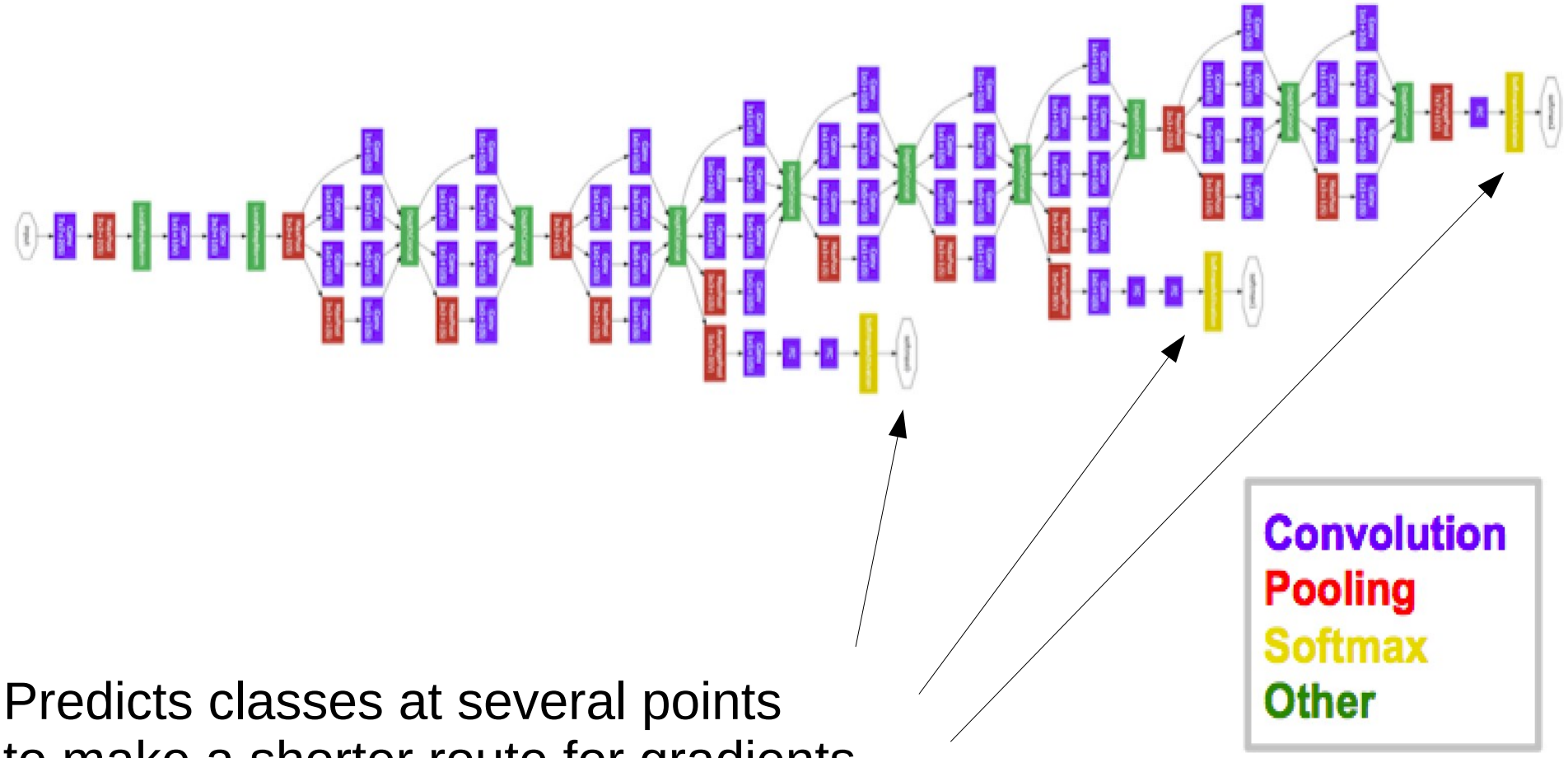


Image taken from Joe Marino, http://joelouismarino.github.io/

Recap: Inception

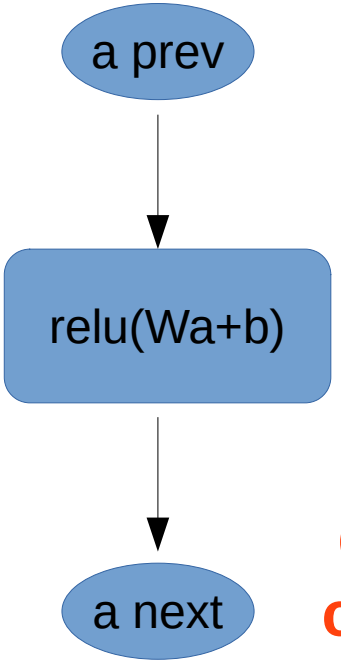


Predicts classes at several points
to make a shorter route for gradients

Recap: Residual Layer

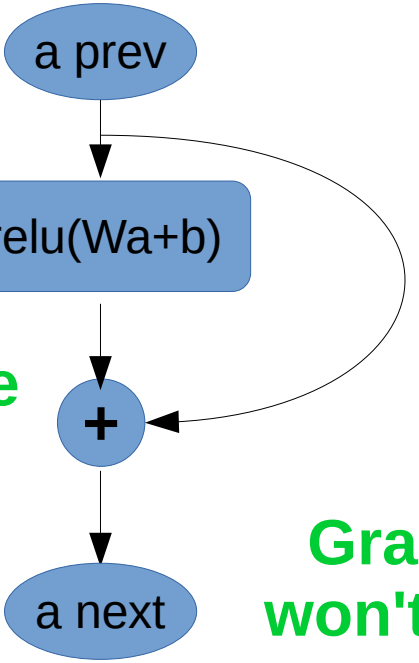
Idea: let's create a shortcut for gradients

Normal layer



Gradients can vanish

Residual layer



elemwise addition

Gradients won't vanish

$$f_{w,b}(x) = relu(W \cdot a + b)$$

$$\nabla f_{w,b}(x) = \nabla relu(W \cdot a + b)$$

$$f_{w,b}(x) = relu(W \cdot a + b) + X$$

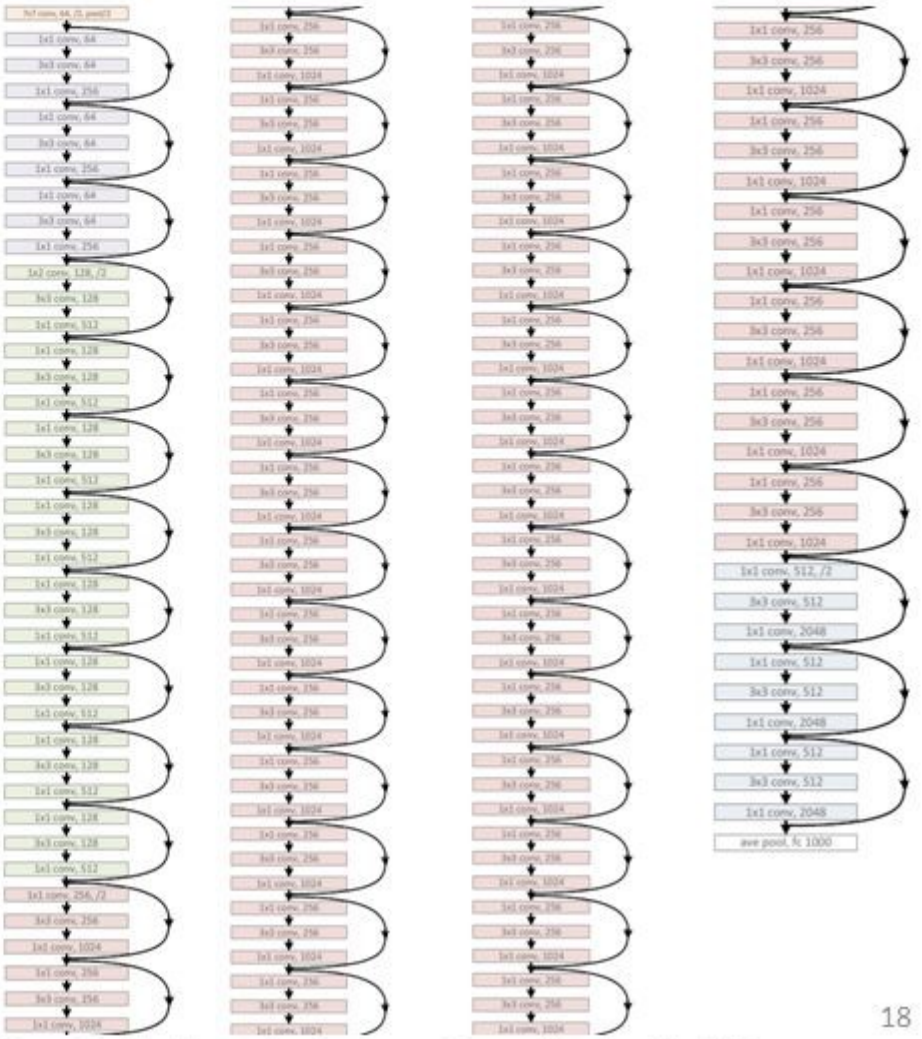
$$\nabla f_{w,b}(x) = \nabla relu(W \cdot a + b) + \vec{1}$$

Recap: Resnet-152

Each layer learns to “modify” previous activation

Much less parameters than vgg16/vgg19

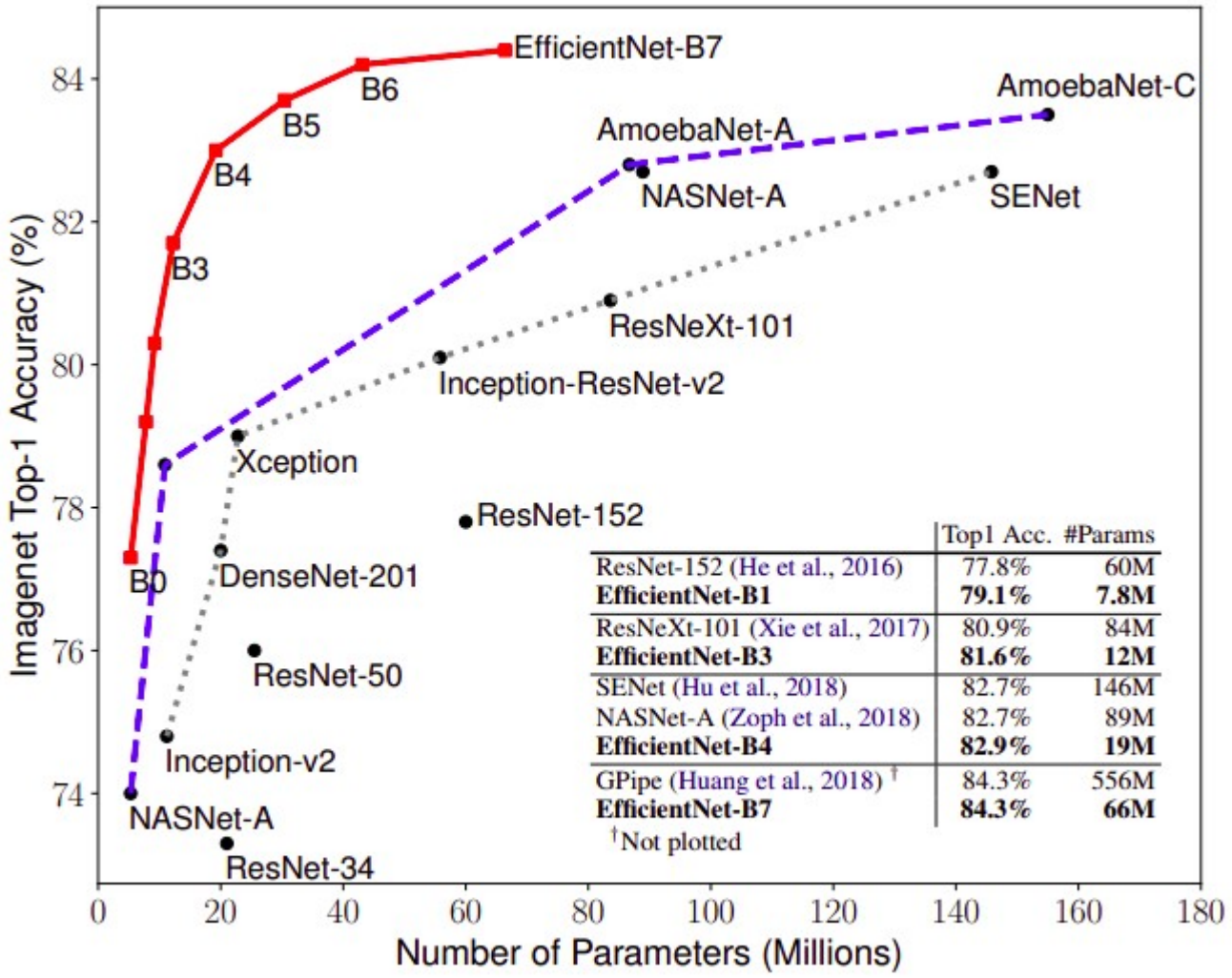
Can be used for other tasks: segmentation, detection, captioning
(just like vgg/inception/...)



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Recap: EfficientNet

https://arxiv.org/abs/1905.11946



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

The only problem: you only have 5k images

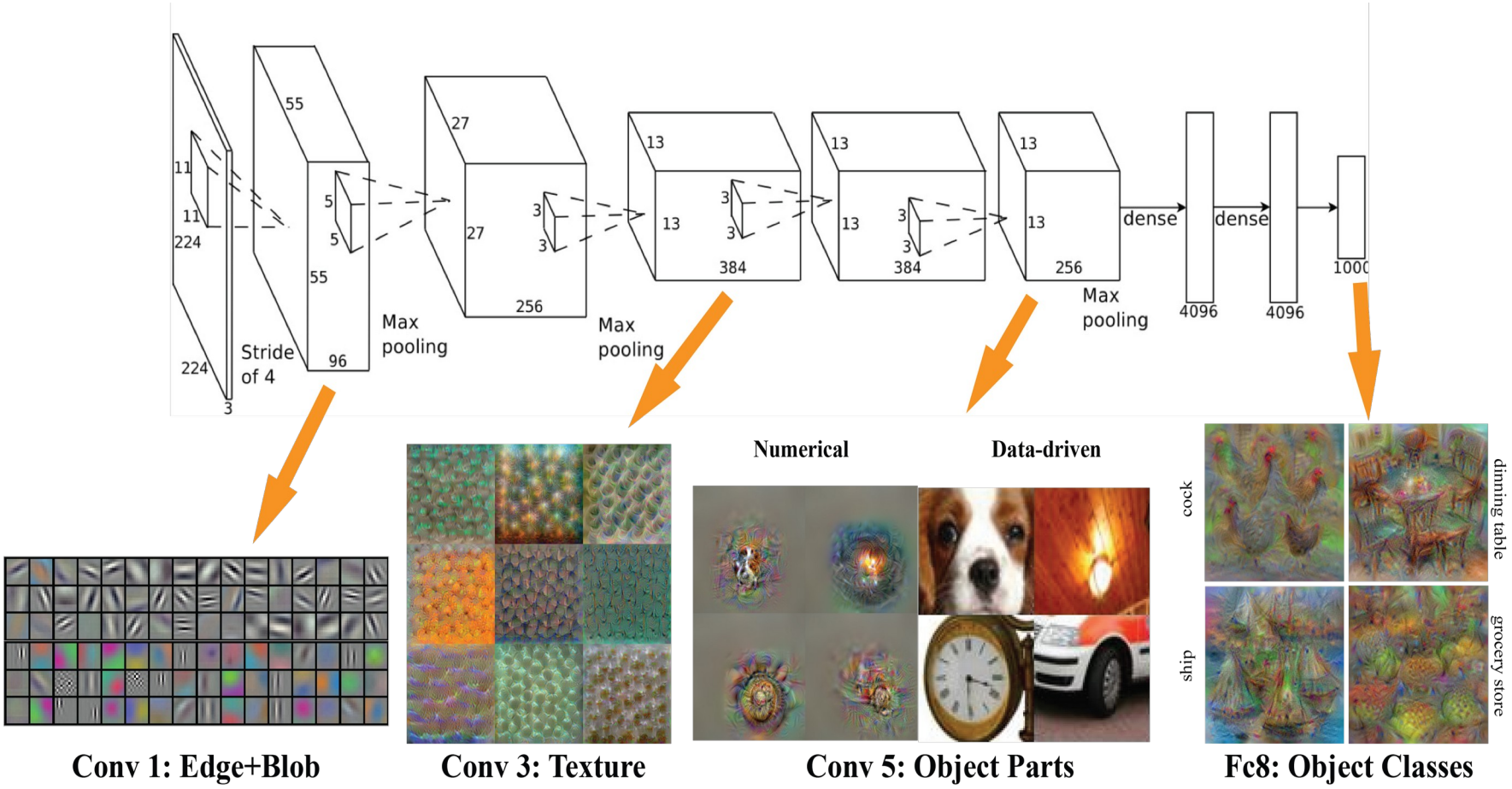
e.g. transport vs military aircrafts

Ideas?

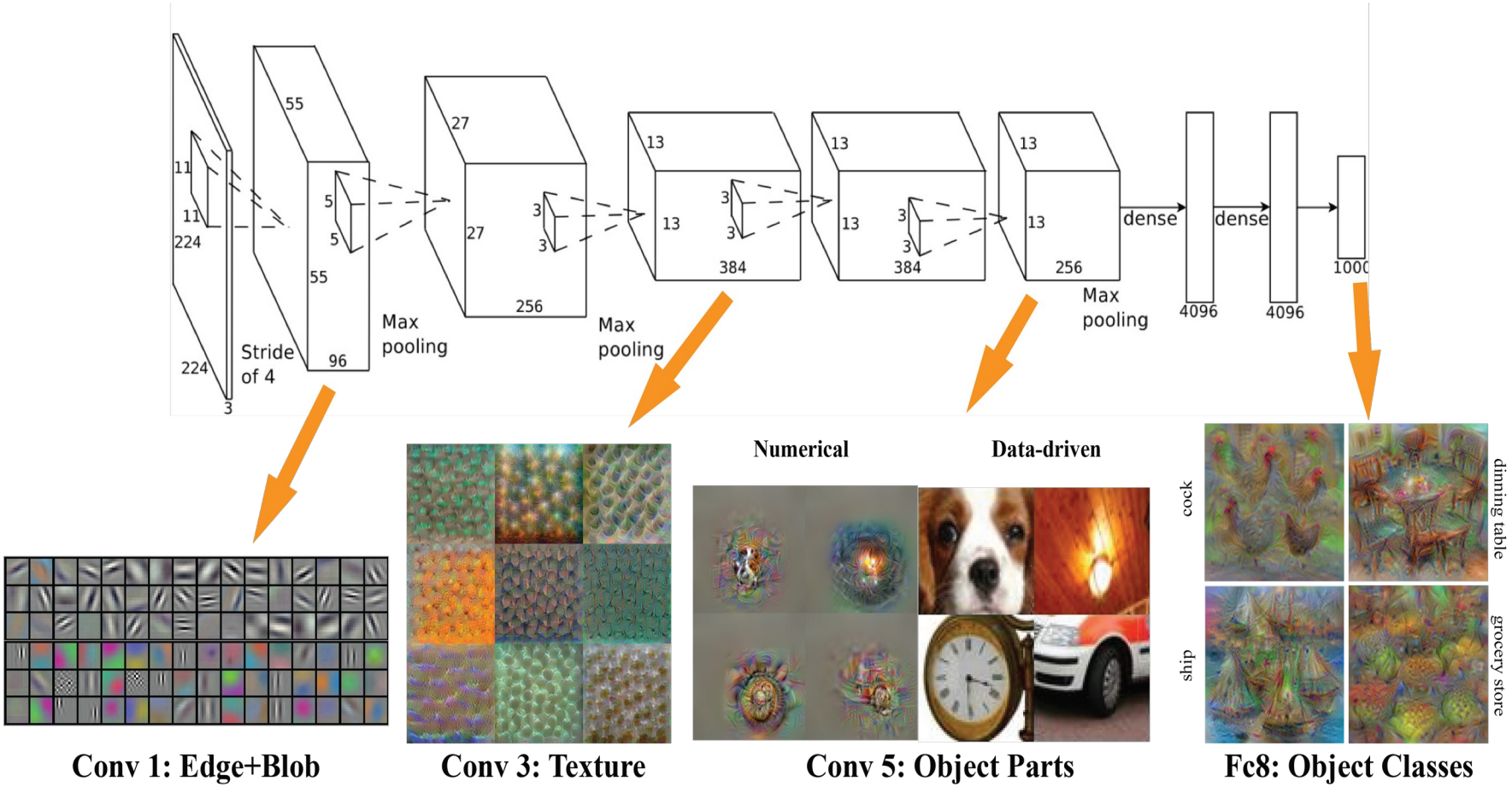
Traditional solutions

- Regularize really hard
- Super small network
- Data augmentation
- Whatever

Feature learning

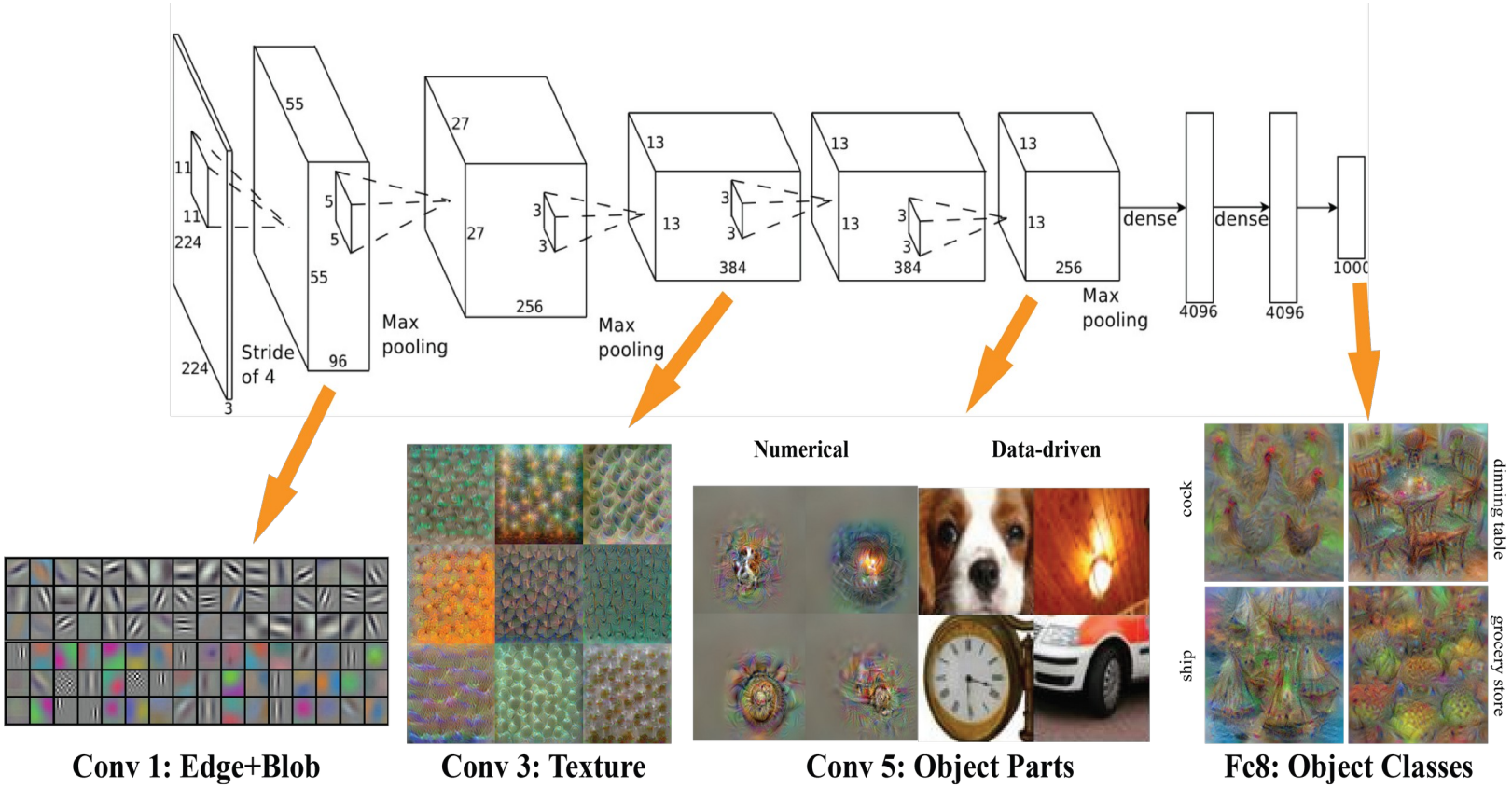


Feature learning



Idea: let's pre-train network on a larger dataset

Can we use them?



~ anywhere
even bran scans

photos
any kinds

casual photos
i.e. not satelite maps

highly similar tasks

Pre-training

- 1. Train a network on large dataset

cifar X

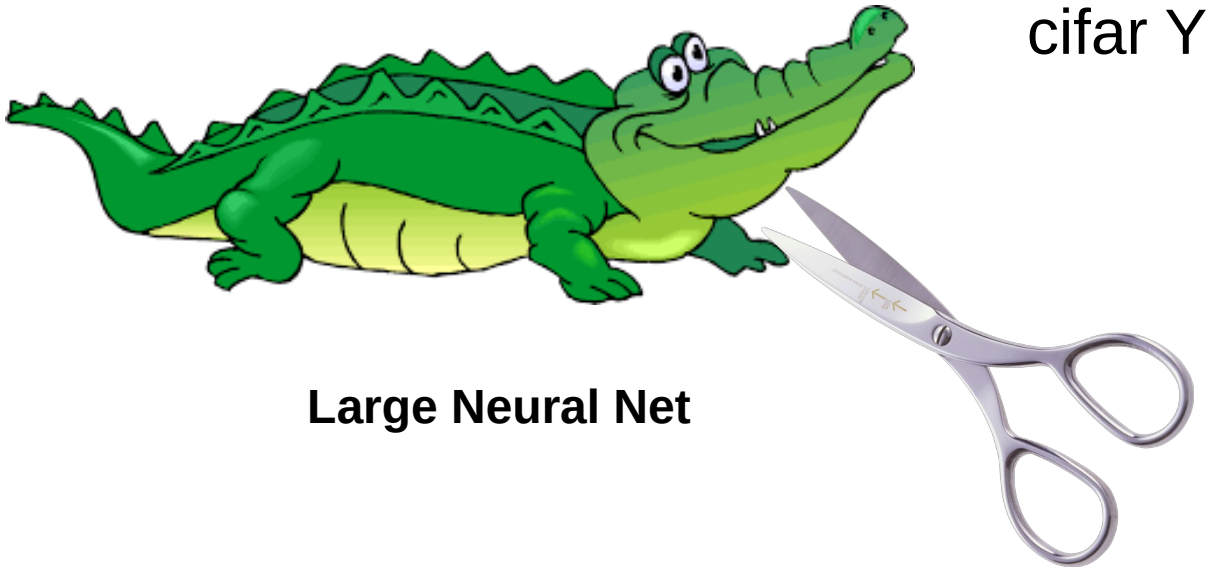
cifar Y

Large Neural Net

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer

cifar X



cifar Y

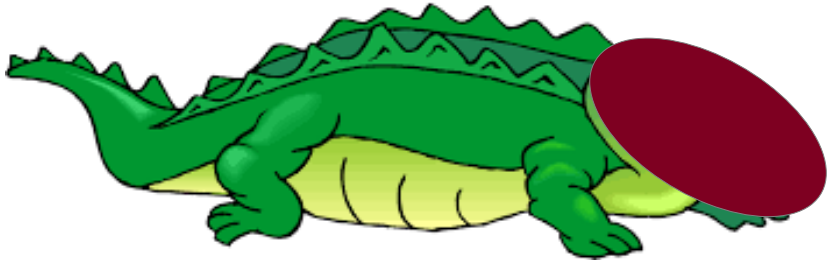
Large Neural Net

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer

cifar Y

cifar X

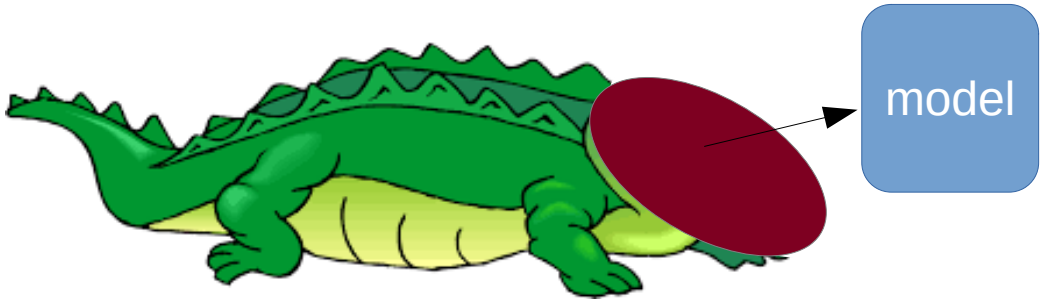


Large Neural Net

Pre-training

- 1. Train a network on large dataset
- 2. Take some intermediate layer
- 3. Build model on top of it

cifar X

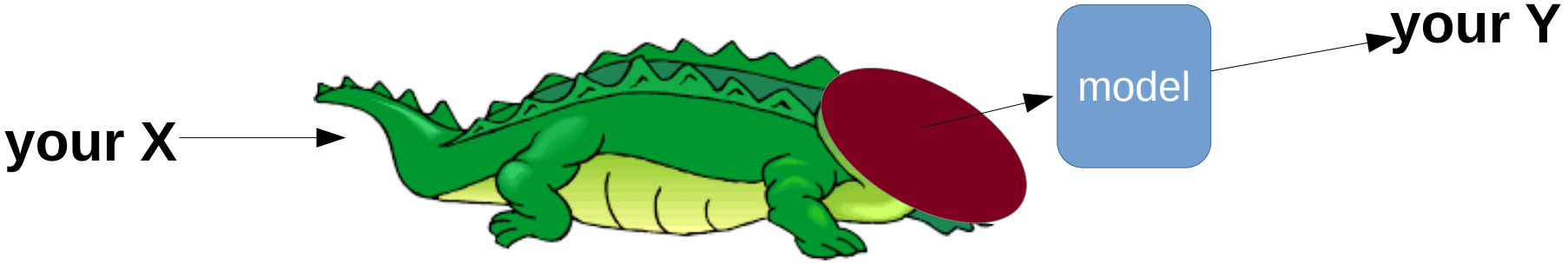


cifar Y

Large Neural Net

Pre-training

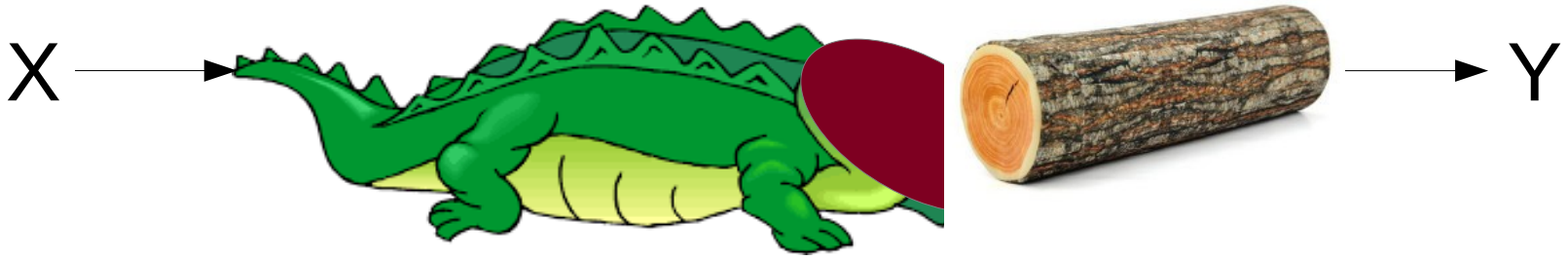
- 1. Train a network on large dataset
- 2. Take some intermediate layer
- 3. Build model on top of it
- 4. Train model for your objective



Large Neural Net

Reusing features

- Chop off “head”
- use “neck” as feature extractor
- Train ANY classifier
  - even random forest will do



Large Neural Net

Fine-tuning

- Chop off “head”
- “freeze” body (consider constant)
- Build new neural network in it's place
- Train “head” only for several iteratons
- Un-freeze body and train full network

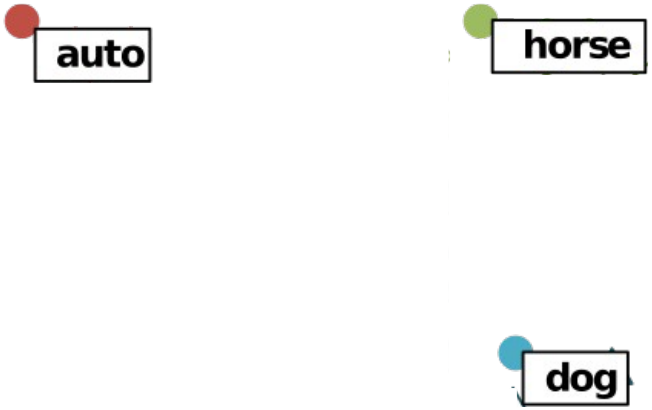
Large Neural Net

Q: What if we only have 1-5 images for each class?

Few-shot learning with KNN

Idea: run KNN over frozen pre-trained model

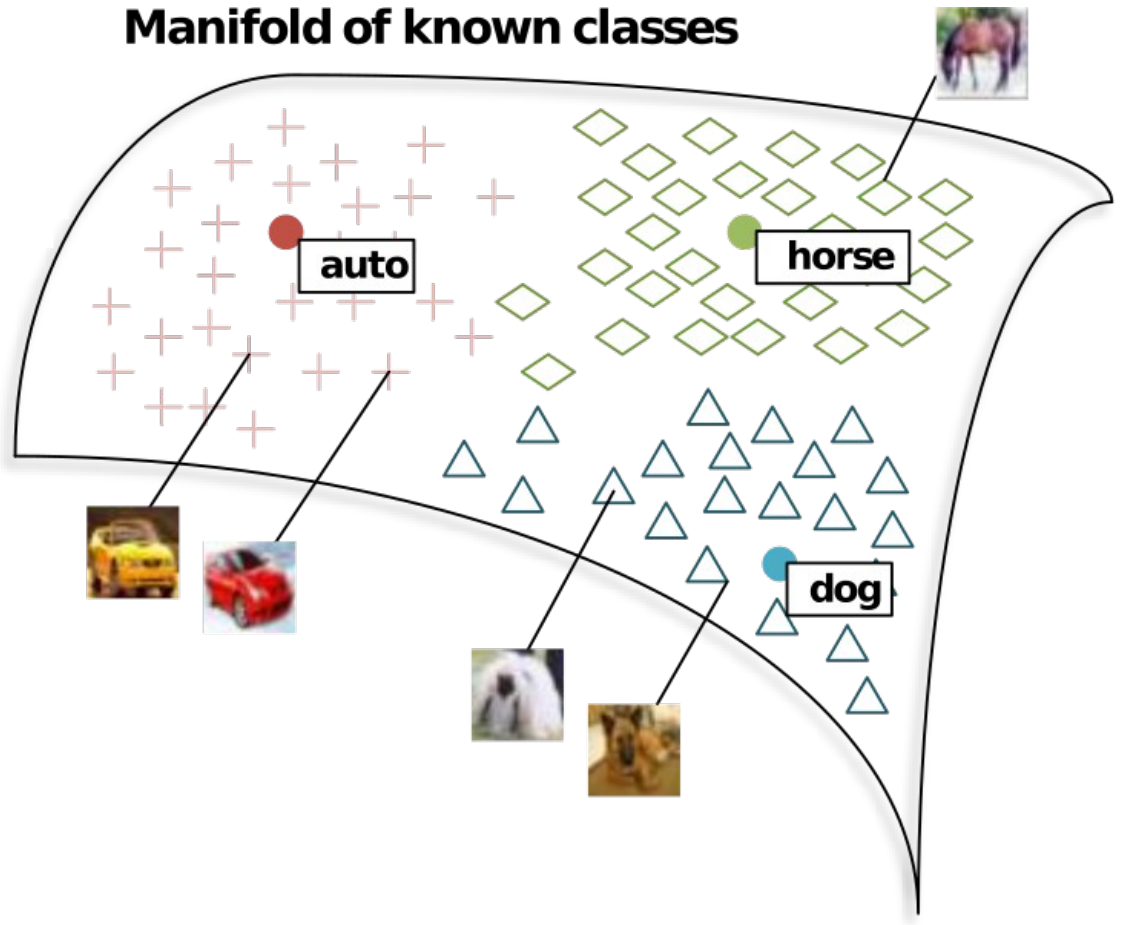
- training images, color = class



Few-shot learning with KNN

Idea: run KNN over frozen pre-trained model

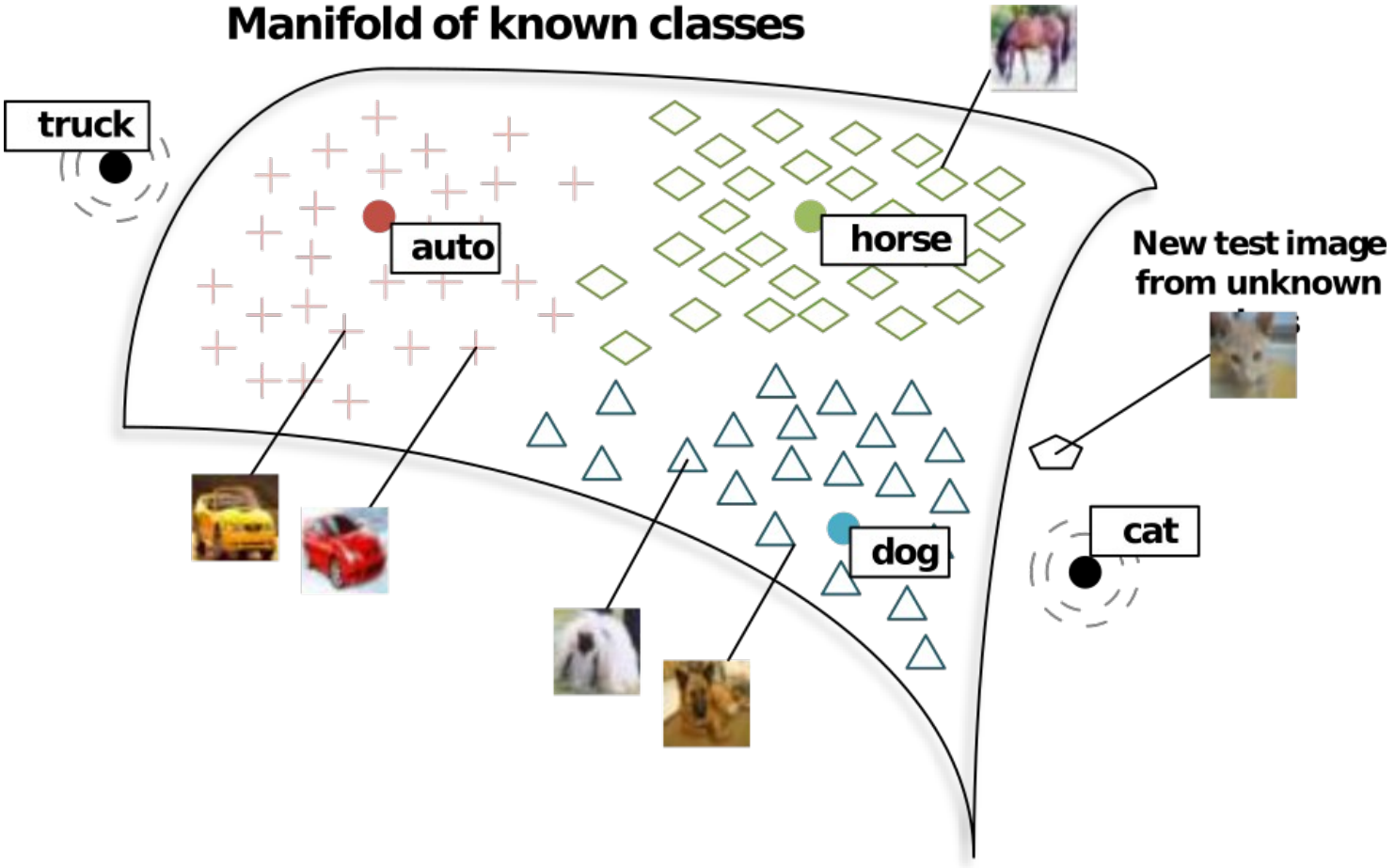
● ● ● - training images, color = class



Few-shot learning with KNN

Idea: run KNN over frozen pre-trained model

● ● ● - training images, color = class



What if we do not have

any labeled dataset?

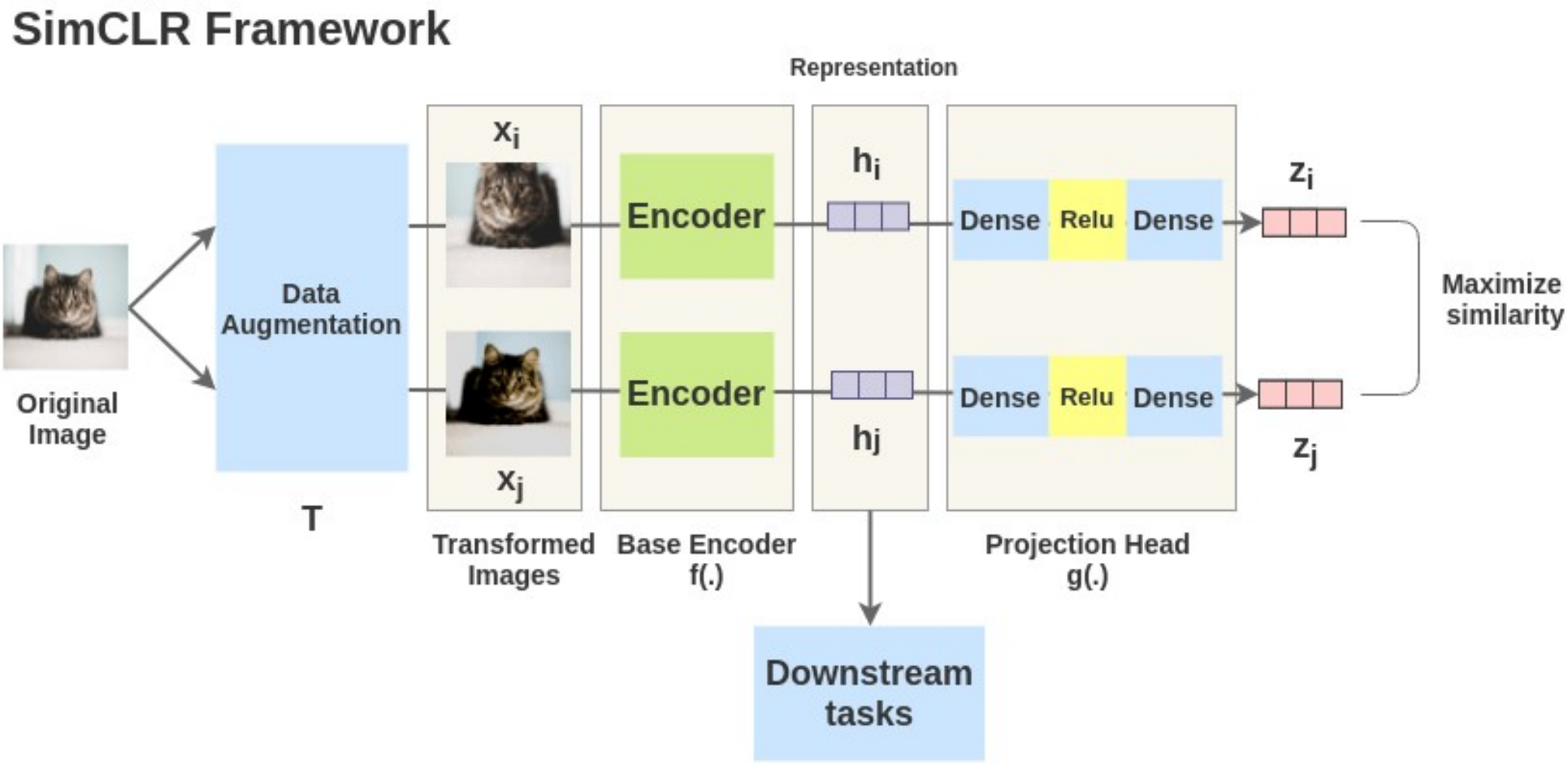
What if we do not have

any ~~labeled~~ dataset?

Can we learn features from just images?

Self-supervised learning

- SimCLR: https://arxiv.org/abs/2002.05709



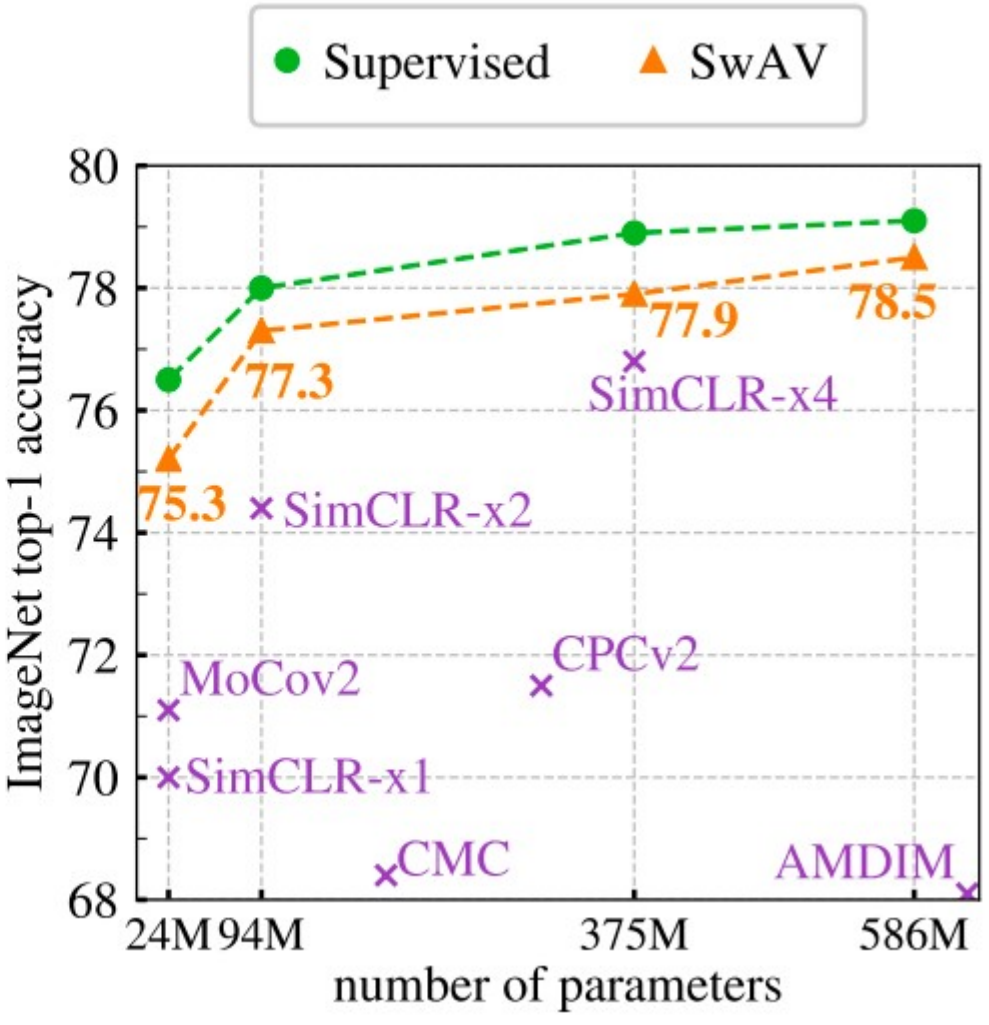
Self-supervised learning

- SwAV: https://arxiv.org/abs/2006.09882

Self-supervised learning

- ImageNet top-1 accuracy: getting close!

Method	Arch.	Param.	Top1
Supervised	R50	24	76.5
Colorization [65]	R50	24	39.6
Jigsaw [46]	R50	24	45.7
NPID [58]	R50	24	54.0
BigBiGAN [15]	R50	24	56.6
LA [68]	R50	24	58.8
NPID++ [44]	R50	24	59.0
MoCo [24]	R50	24	60.6
SeLa [2]	R50	24	61.5
PIRL [44]	R50	24	63.6
CPC v2 [28]	R50	24	63.8
PCL [37]	R50	24	65.9
SimCLR [10]	R50	24	70.0
MoCov2 [11]	R50	24	71.1
SwAV	R50	24	**75.3**

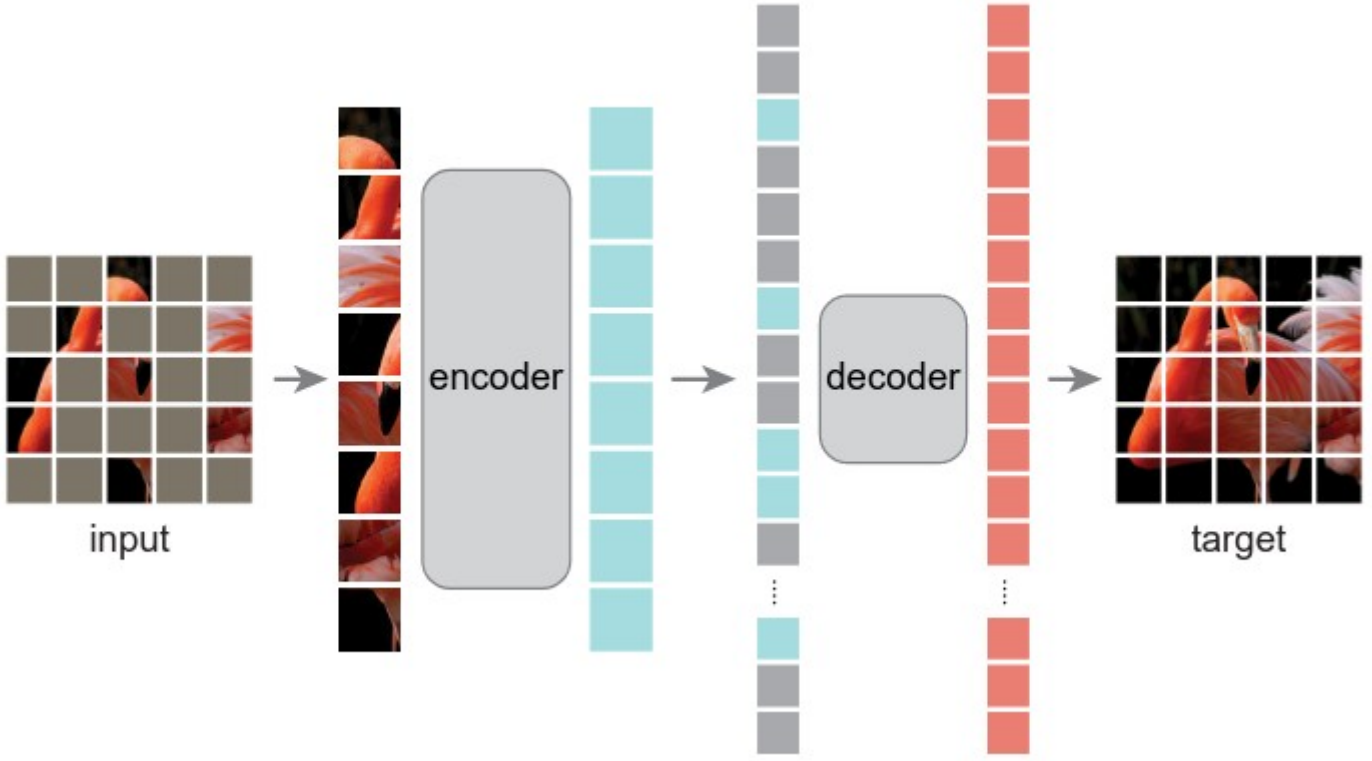


Masked Image Modeling

https://arxiv.org/abs/2111.09886
https://arxiv.org/abs/2111.06377

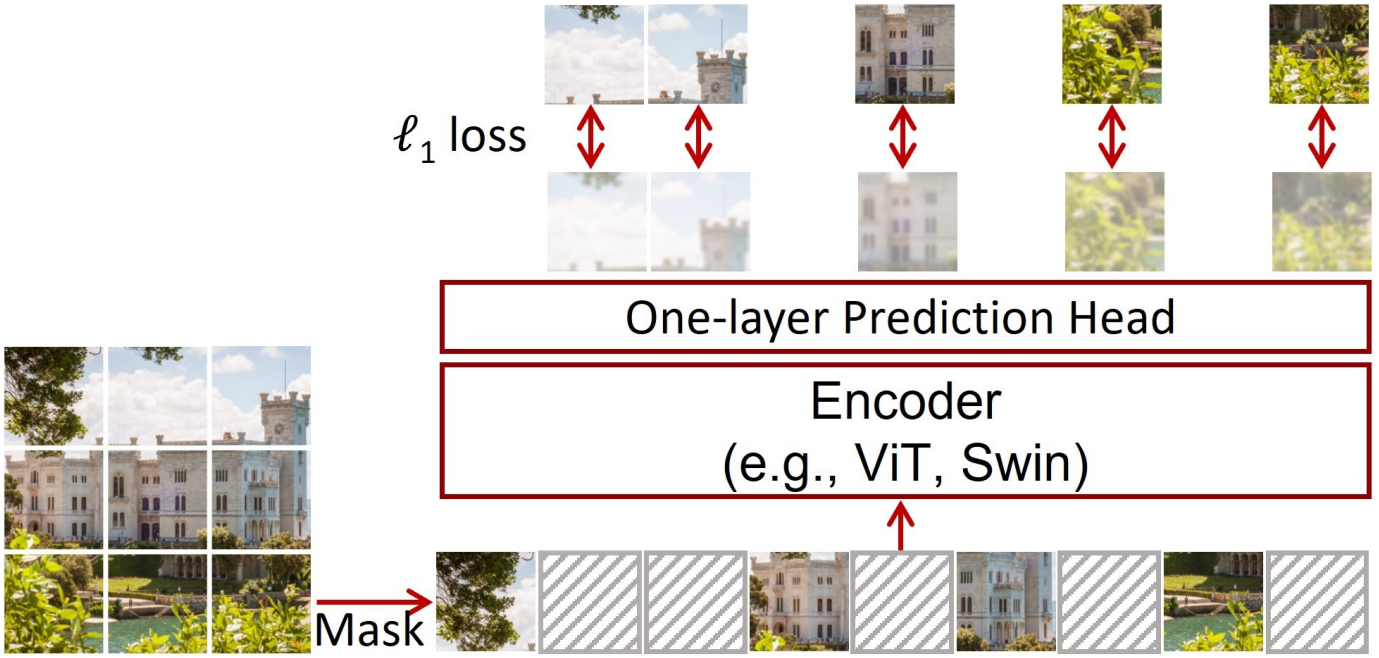
- Remove random tiles, try to reconstruct them!

  reuse the pre-trained network for other tasks



Masked Image Modeling

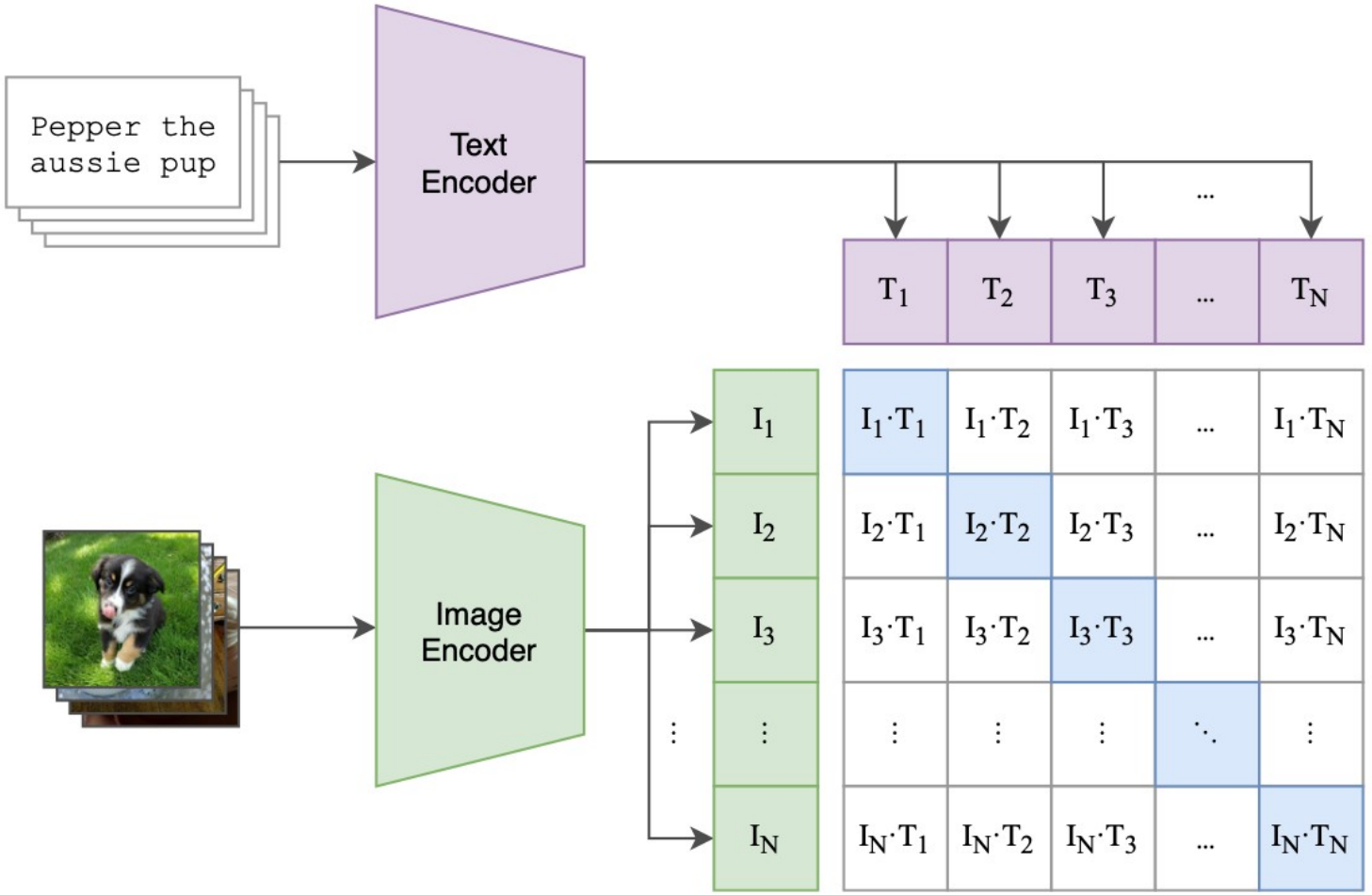
https://arxiv.org/abs/2111.09886
https://arxiv.org/abs/2111.06377



Side-quest: CLIP

https://arxiv.org/abs/2103.00020

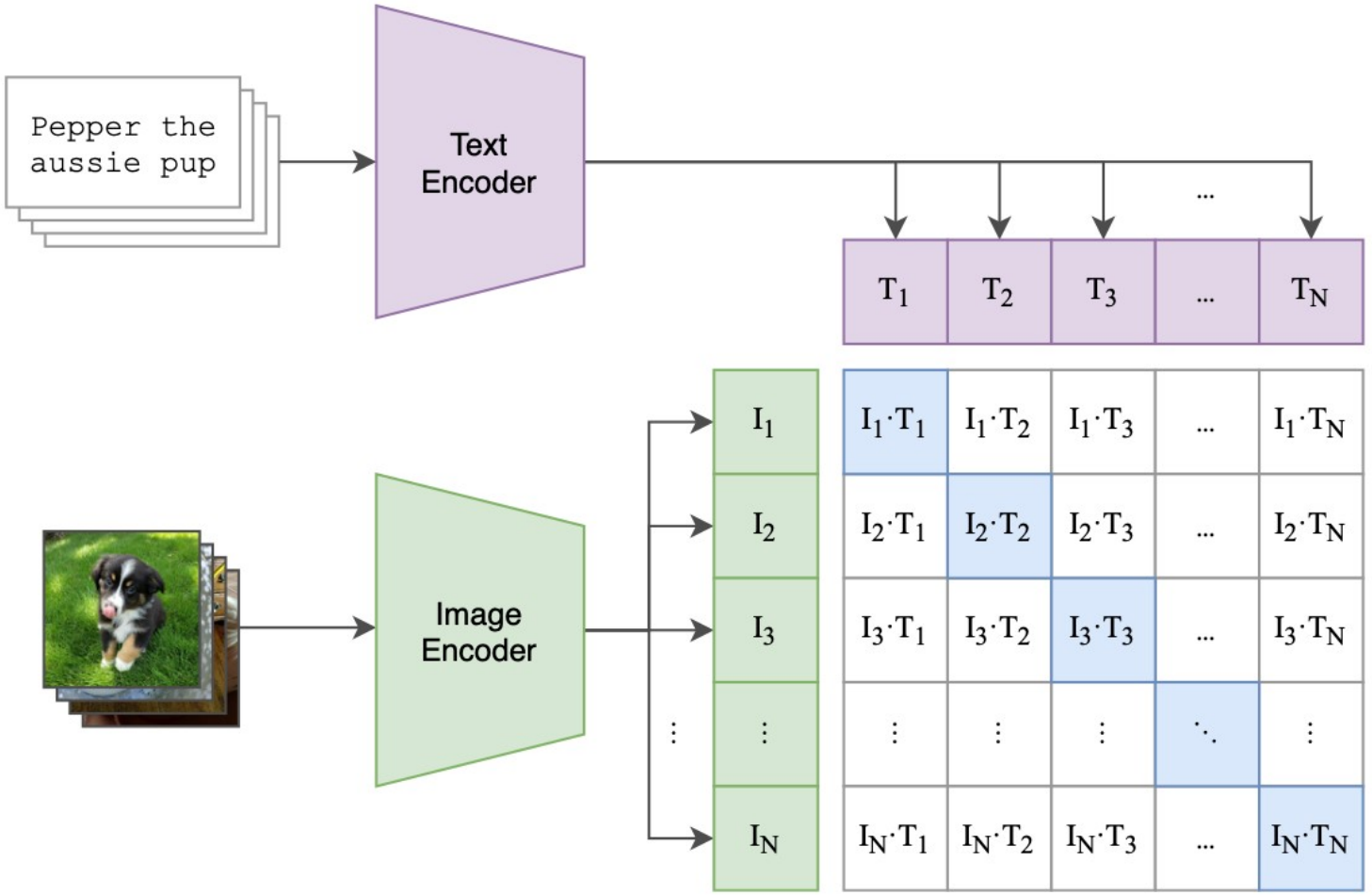
Data: a large dataset of images with text captions
Objective: image and it's caption must have close vectors
different images/captions should not have close vectors



Side-quest: CLIP

https://arxiv.org/abs/2103.00020

Data: a large dataset of images with text captions
Loss function: contrastive, similar to SimCLR

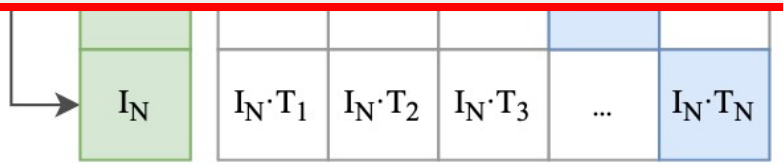


Side-quest: CLIP

https://arxiv.org/abs/2103.00020

Data: a large dataset of images with text captions
Loss function: contrastive, similar to SimCLR

More on CLIP later in the course!



What’s the best method?

There are (small) improvements every year...

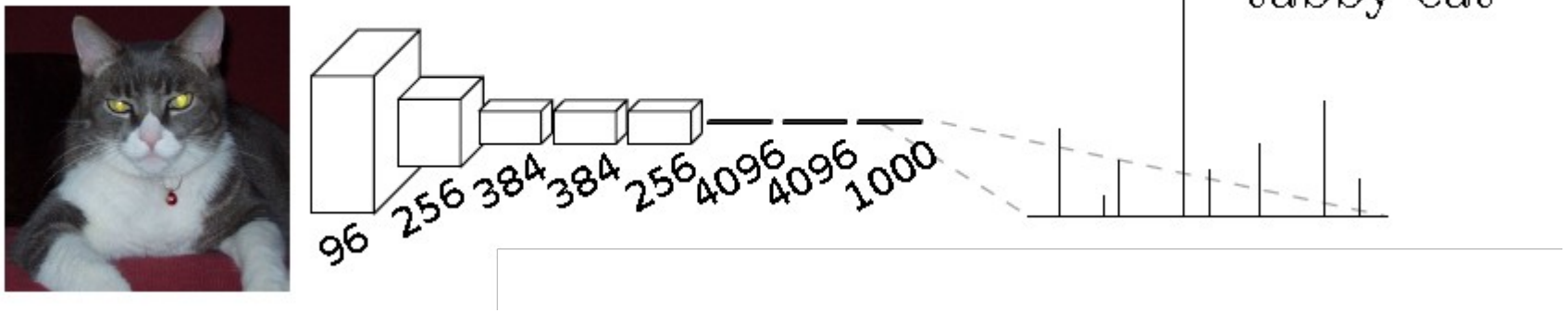
Let’s go look at paperswithcode :)

Alright, we did it!

What else can we do besides image classification?

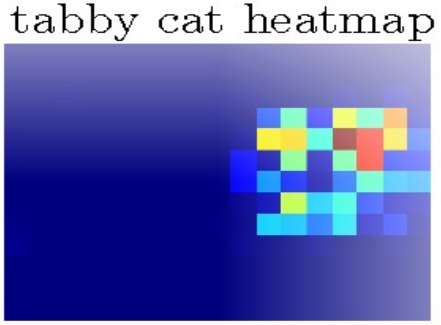
Semantic segmentation

- Classification



- Segmentation/Detection

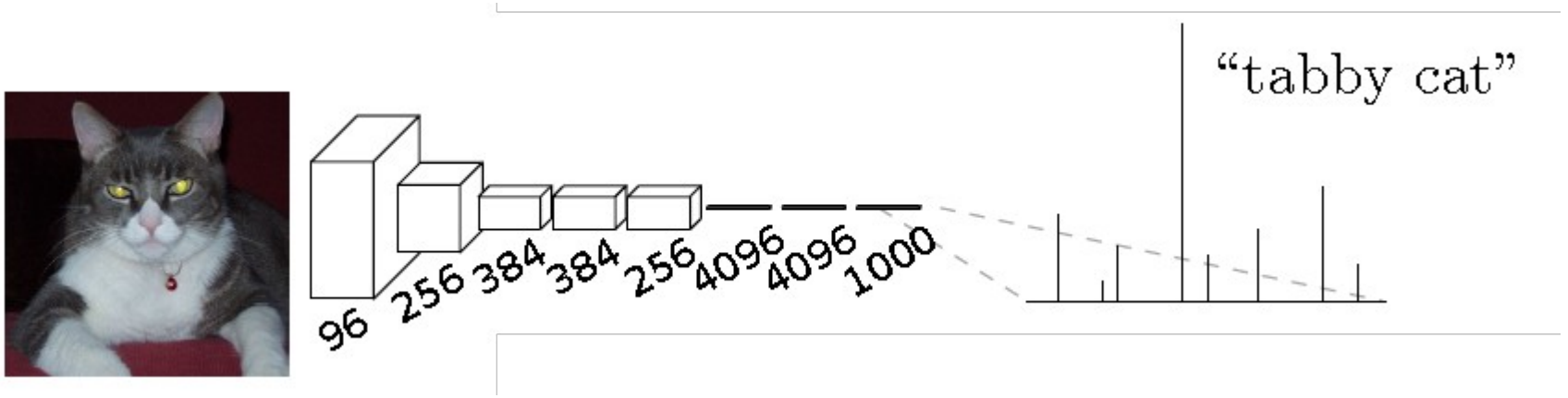
???
Ideas on how to predict that?



- Pics: V. Lempitsky

Semantic segmentation

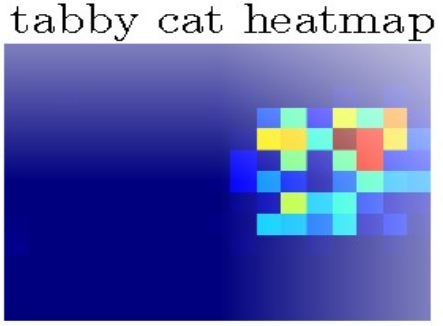
- Classification



- Segmentation/Detection (naïve)

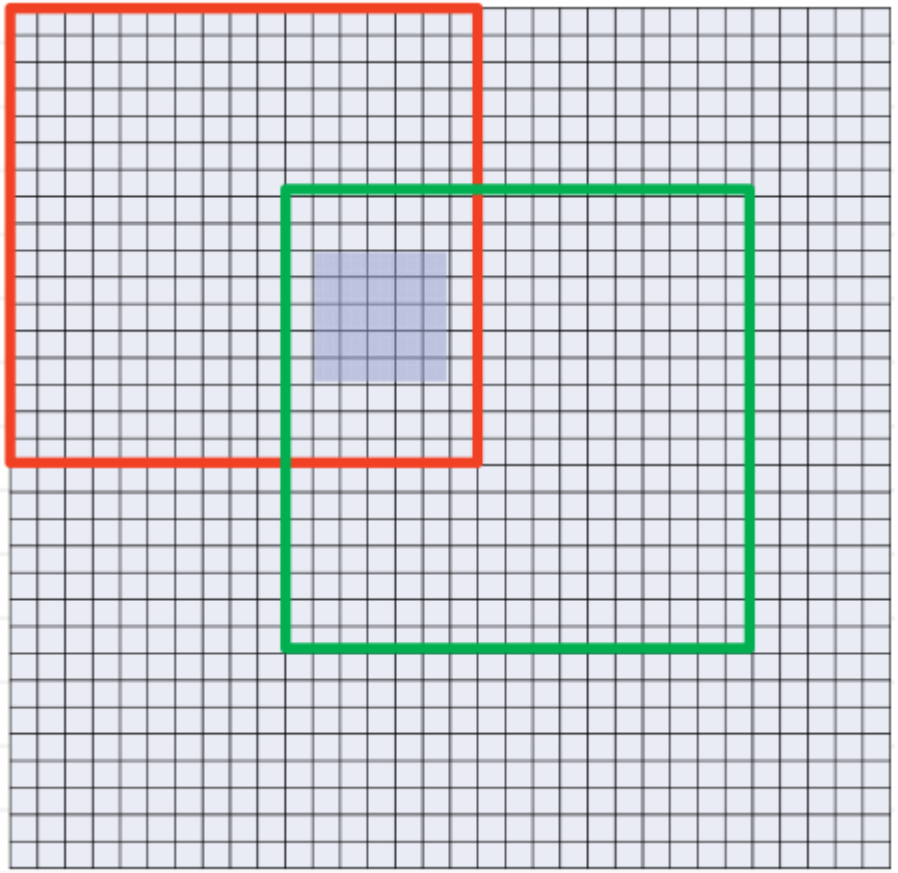
Apply to each spot

"convolution with whole NN as a filter"



- Pics: V. Lempitsky

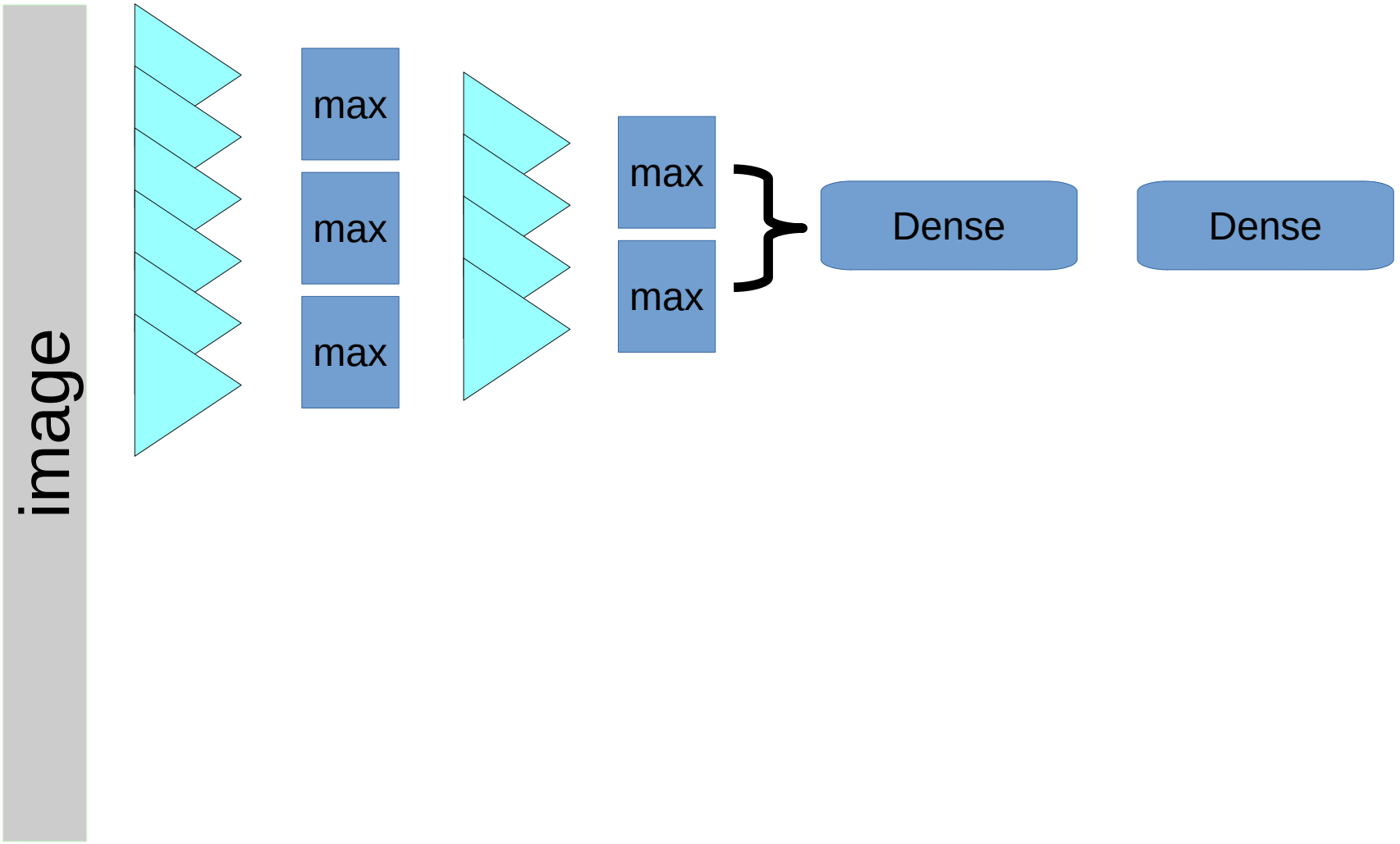
Convolutionalization



- Idea: if you apply the NN to neighboring patches, you'll have to compute same convolutions again
- Instead let us precompute convolutions for the whole image in advance.

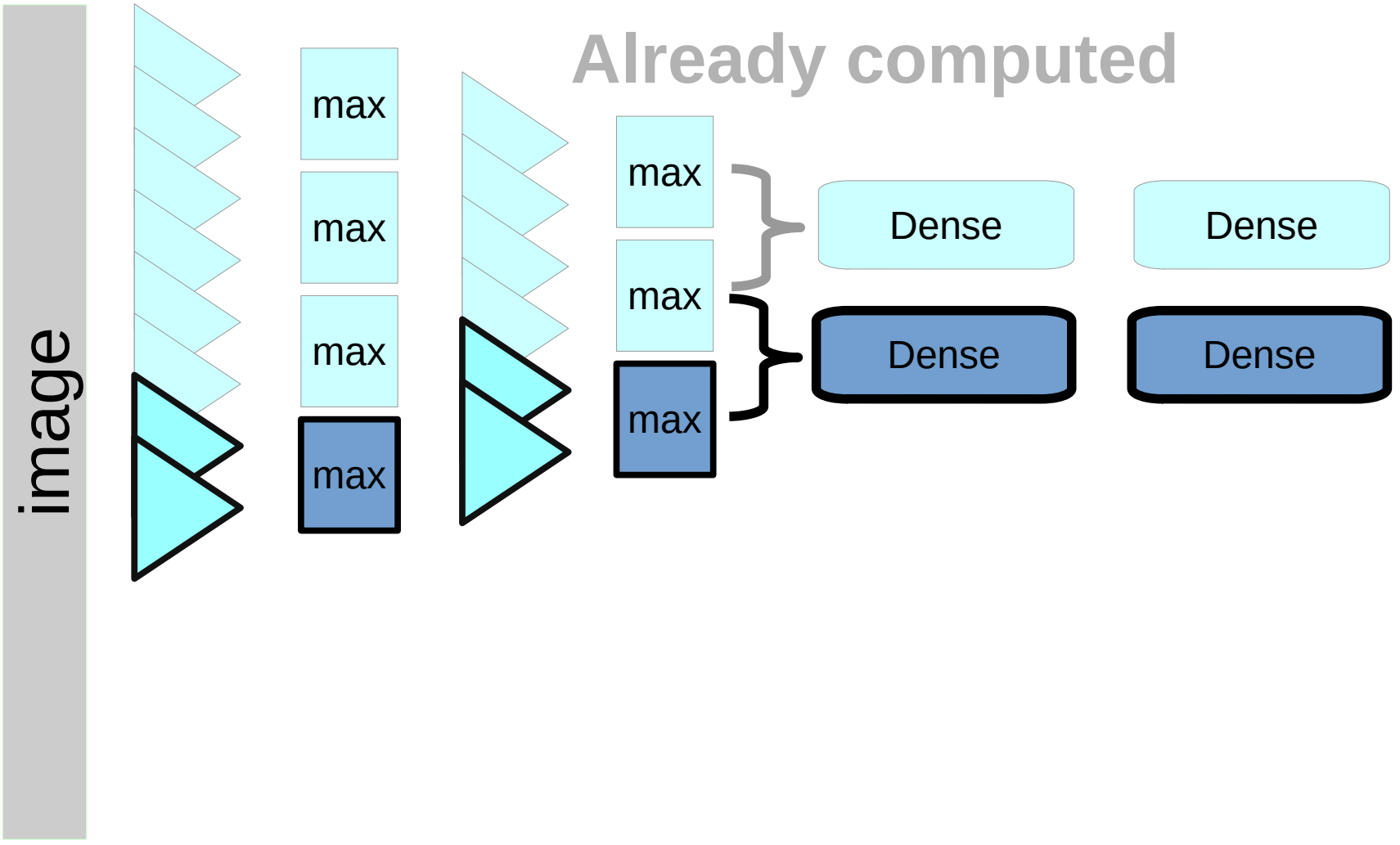
- Pics: V. Lempitsky

Convolutionalization



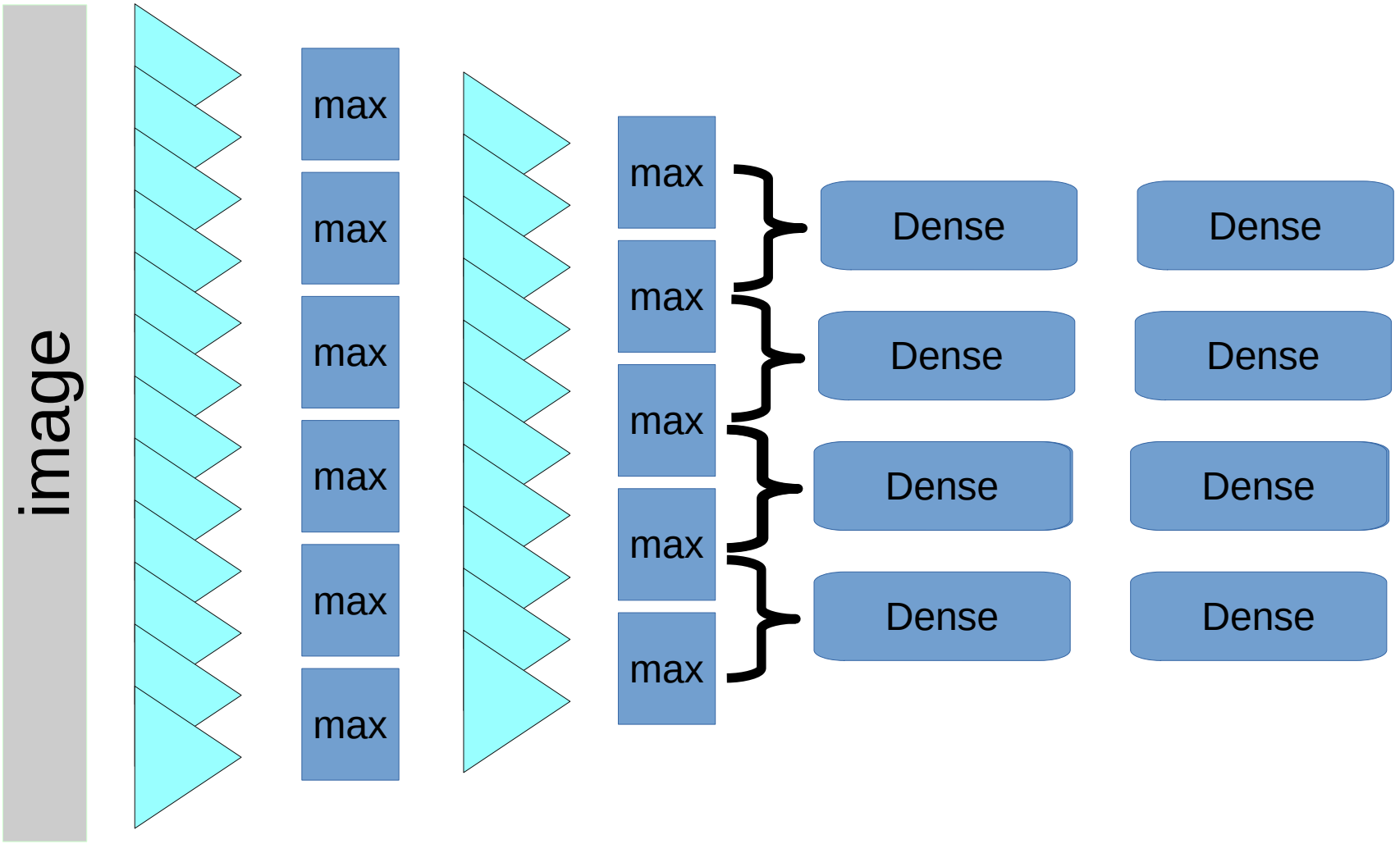
X conv pool conv pool dense output

Convolutionalization



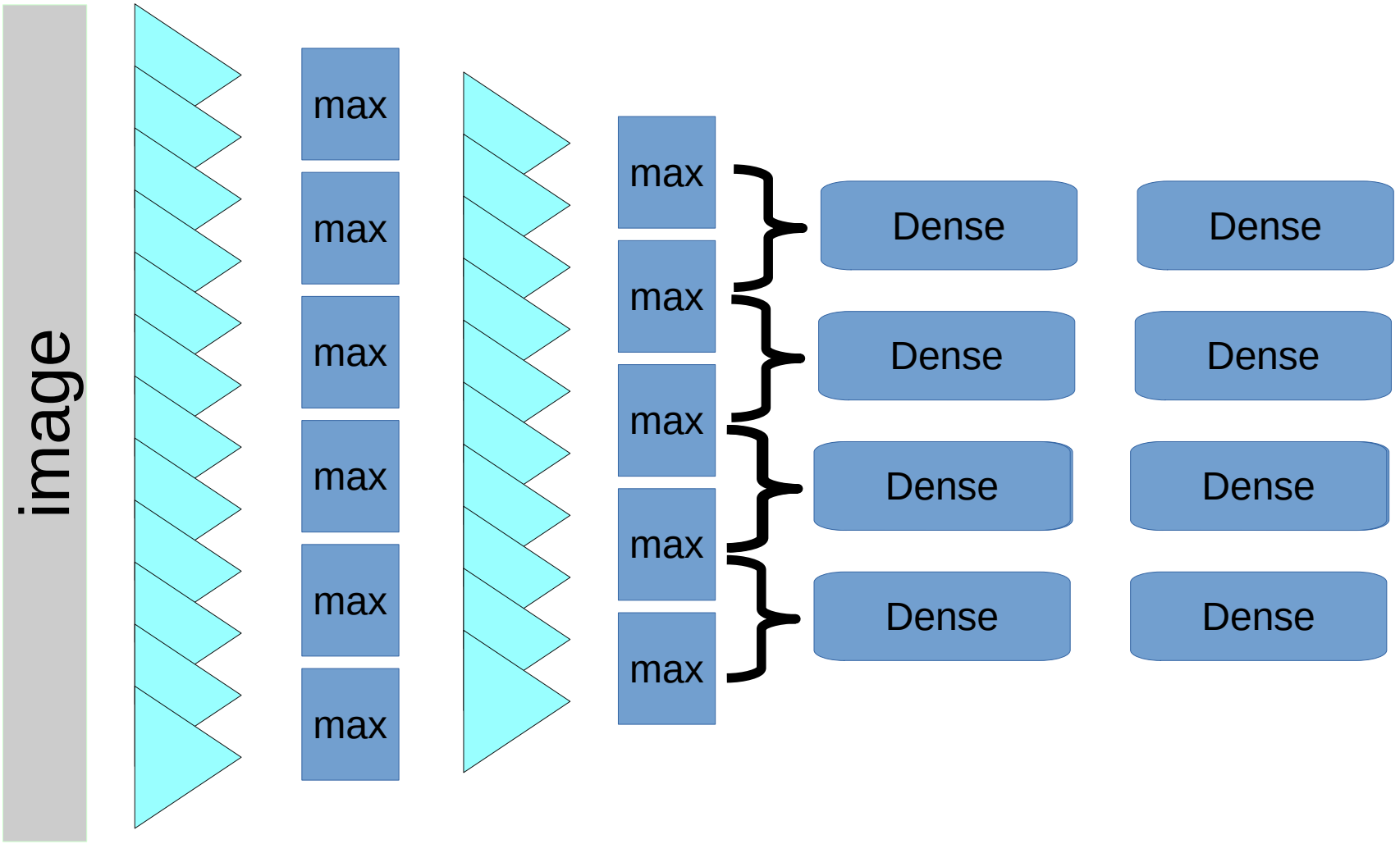
X conv pool conv pool dense output

Convolutionalization



X conv pool conv pool **???** **???**

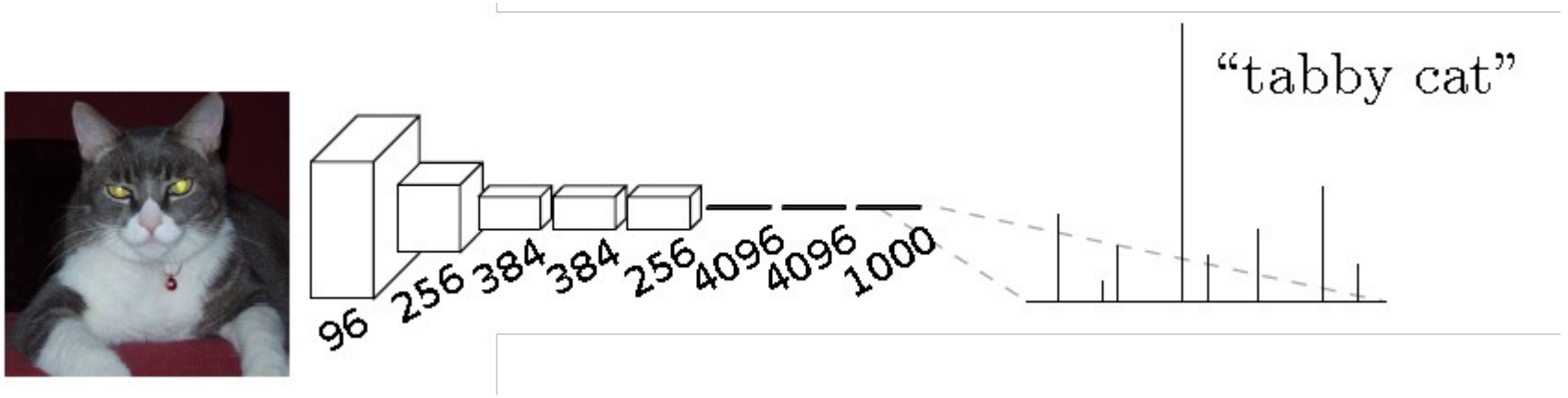
Convolutionalization



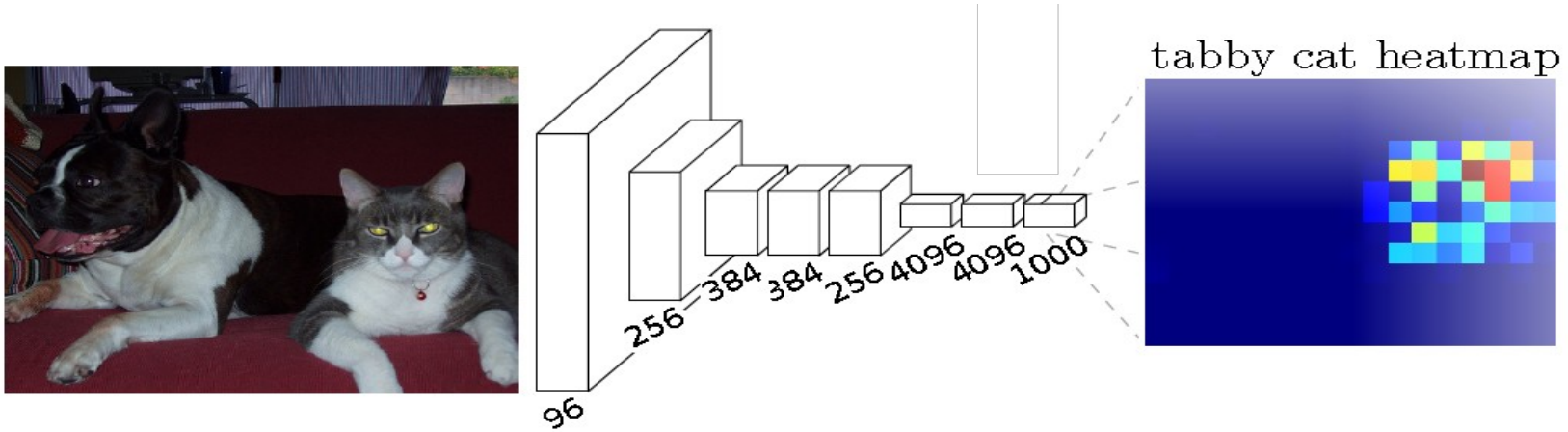
X conv pool conv pool conv2x2 conv1x1

Semantic segmentation

- Classification



- Segmentation/Detection



- Pics: V. Lempitsky

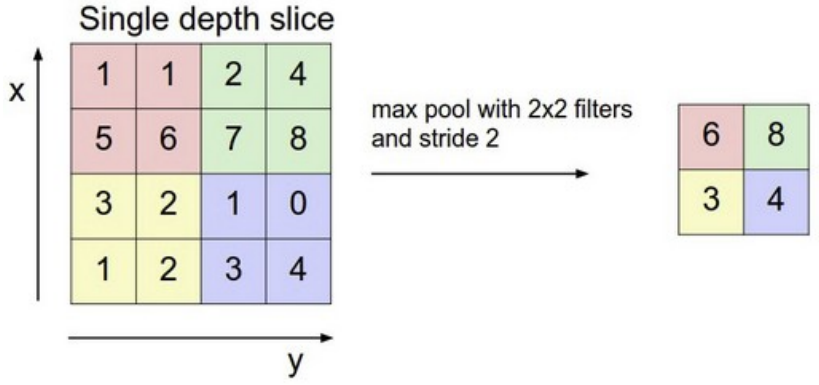
What if output must have same resolution as in the input image? Or even bigger?

How can we **increase** activation size?

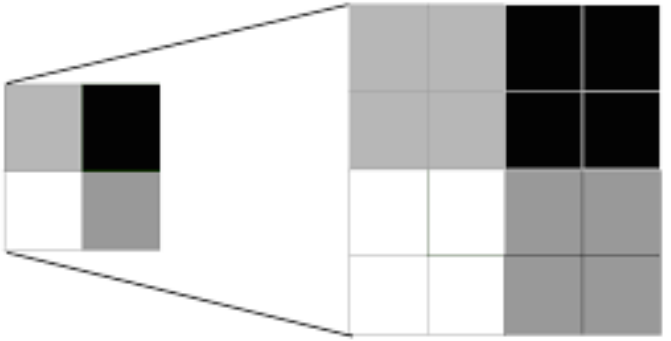
Ideas?

Upscale

- Pooling =
  - take (n*m) region,
  - output one / channel

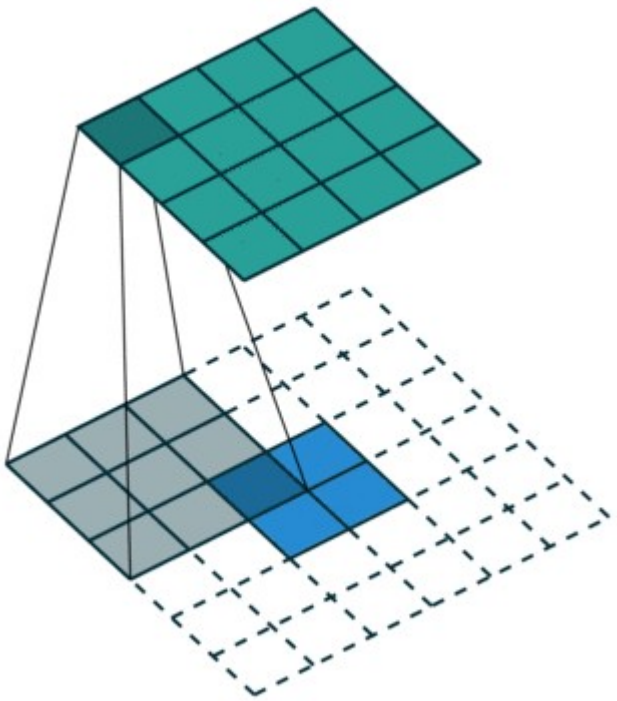


- Upscaling / Unpooling / Upsampling
  - Take one value
  - Output (n*m) region
  - Several strategies



Deconvolution

- The “inverse” of convolution
  - = transposed convolution
  - ~ wide convolution
- deconv(conv(img))
- preserves image shape
- Stride ~ upscale

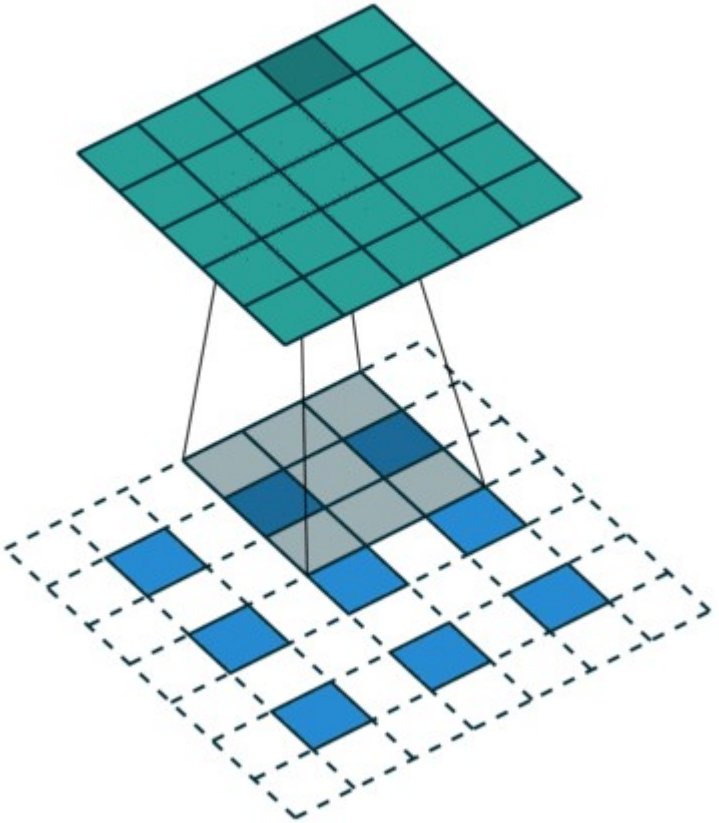


deconvolution

https://github.com/vdumoulin/conv_arithmetic

Deconvolution

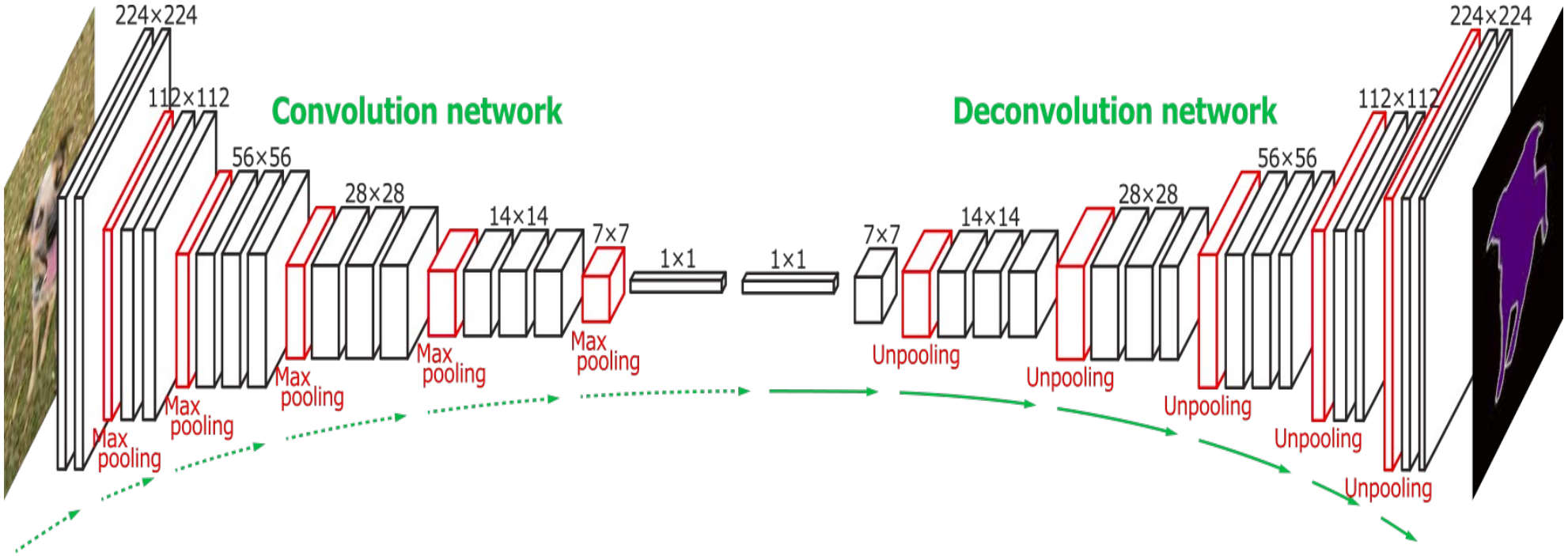
- The “inverse” of convolution
  - = transposed convolution
  - ~ wide convolution
- deconv(conv(img))
- preserves image shape
- Stride ~ upscale



strided deconvolution

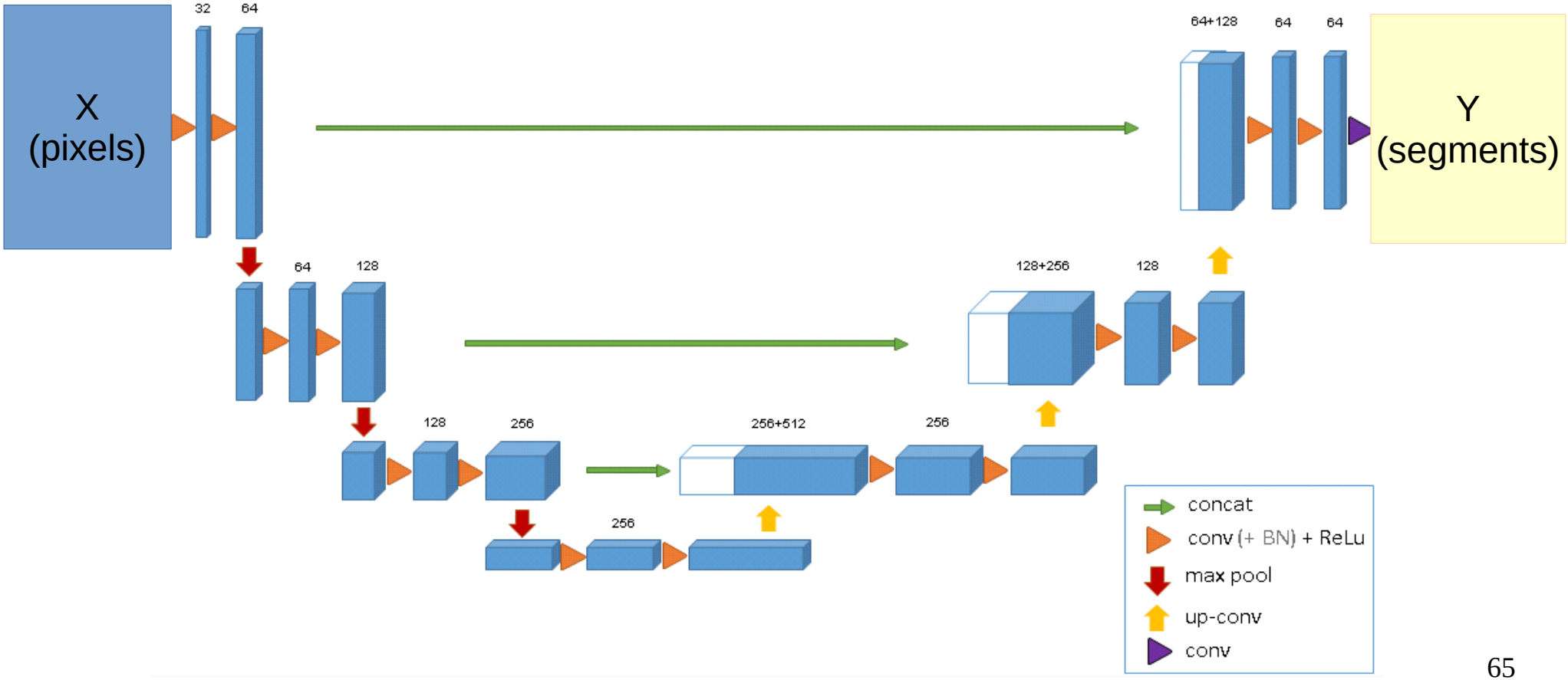
https://github.com/vdumoulin/conv_arithmetic

Fully-convolutional

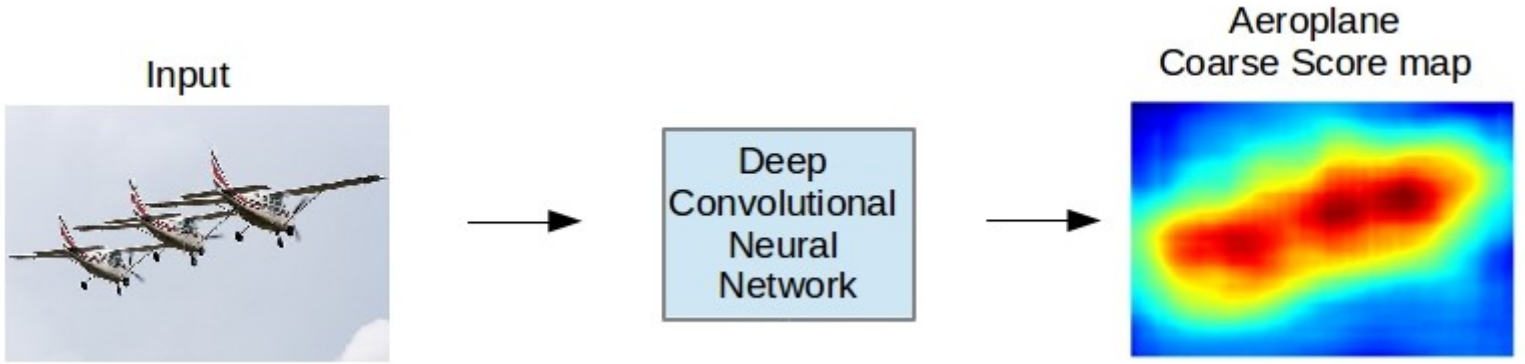


The U-net

- Add connections between layers of similar abstraction
- The idea is similar to gaussian/laplacian pyramids



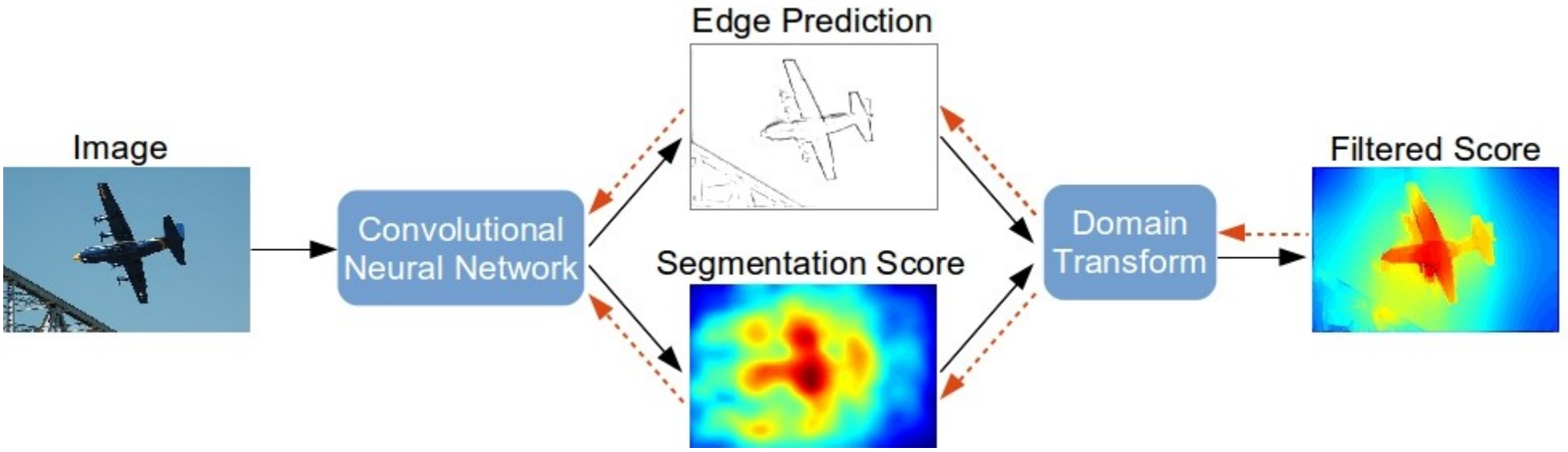
Raw network output is usually rather blurred



How can we get sharper edges?

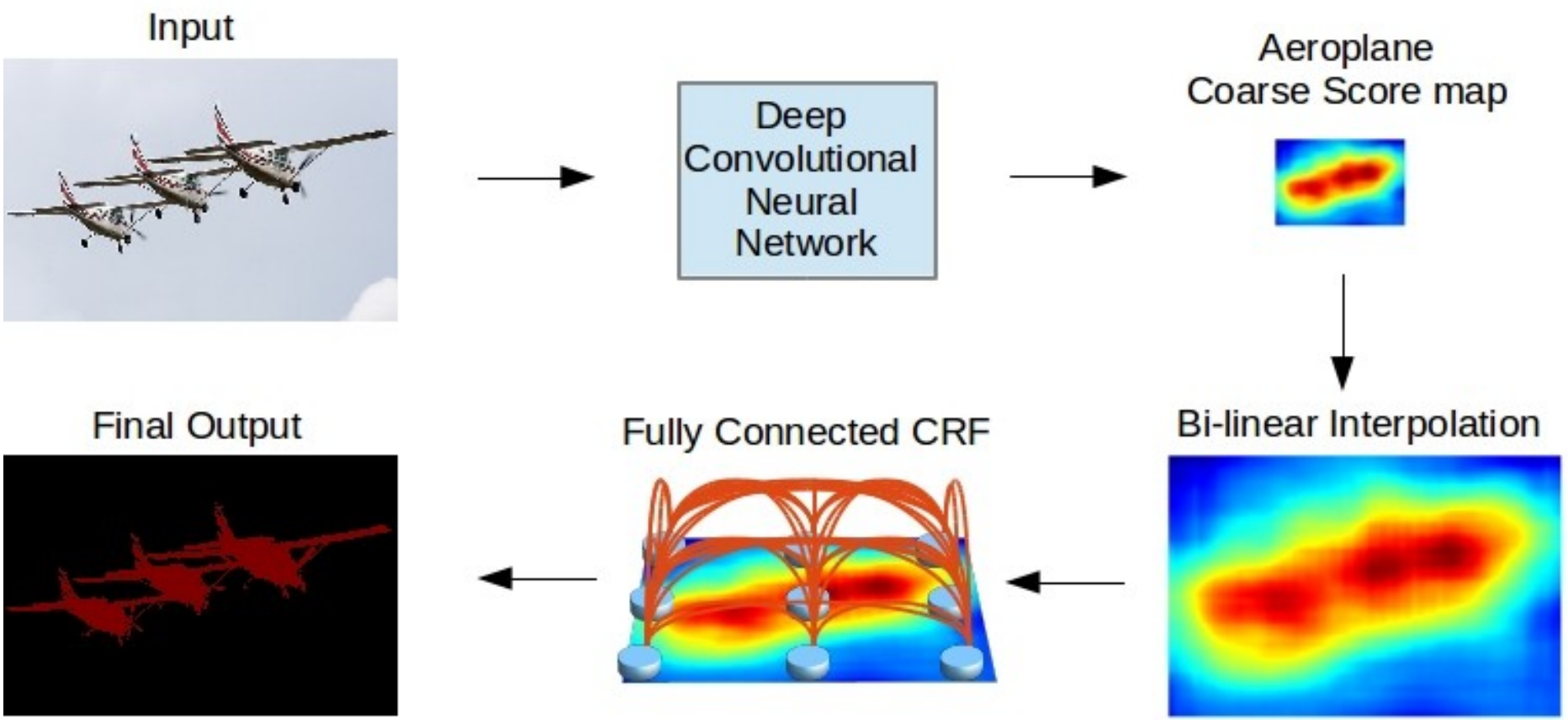
Segmentation: Edge prediction

- Train both class predictor and edge predictor



Segmentation: Graph models

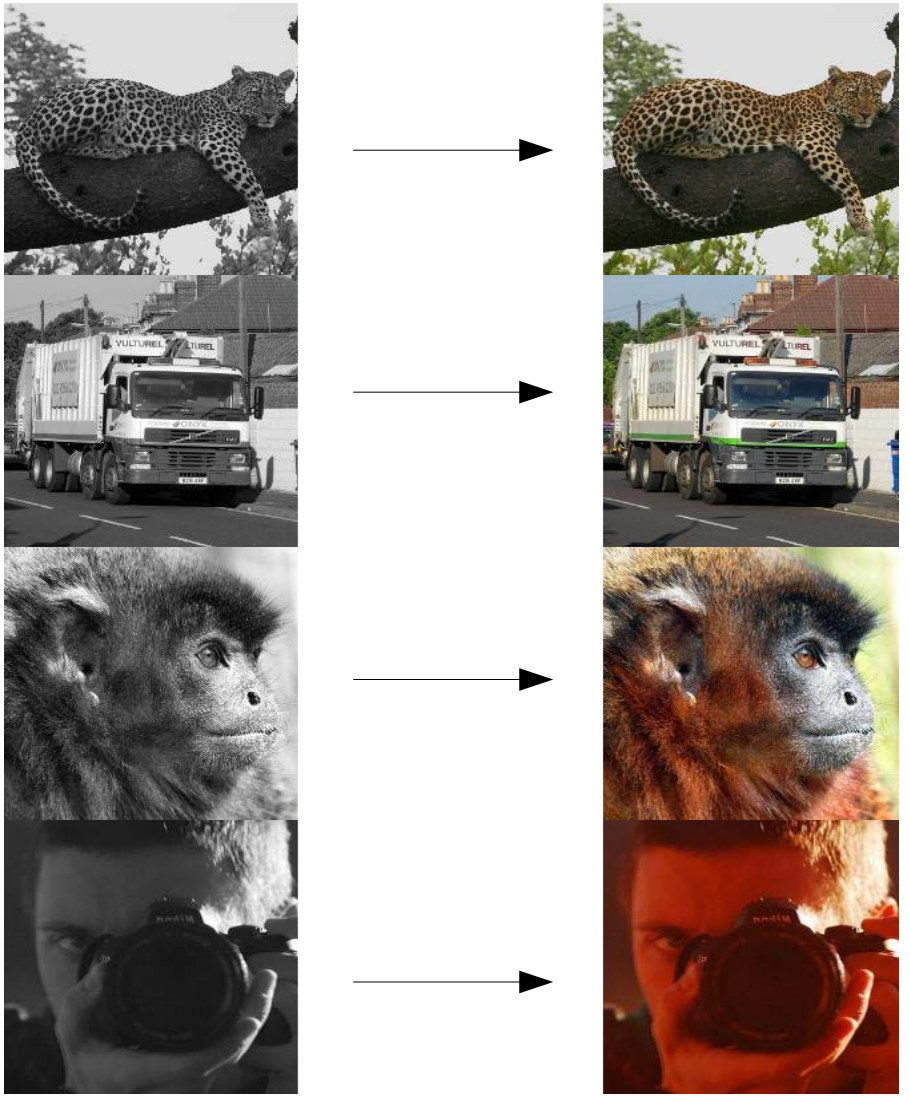
- Tldr incentivize predicting same class over same texture / different classes over edges



See Vetrov's course or https://www.youtube.com/watch?v=uAsys22y5mY

Image coloring

- Input:
  - gray/sepia image
  - mb neighboring frames
- Output:
  - RGB image
  - Same size as input
- **Ideas?**
  - Data?
  - Loss?
  - Architecture?

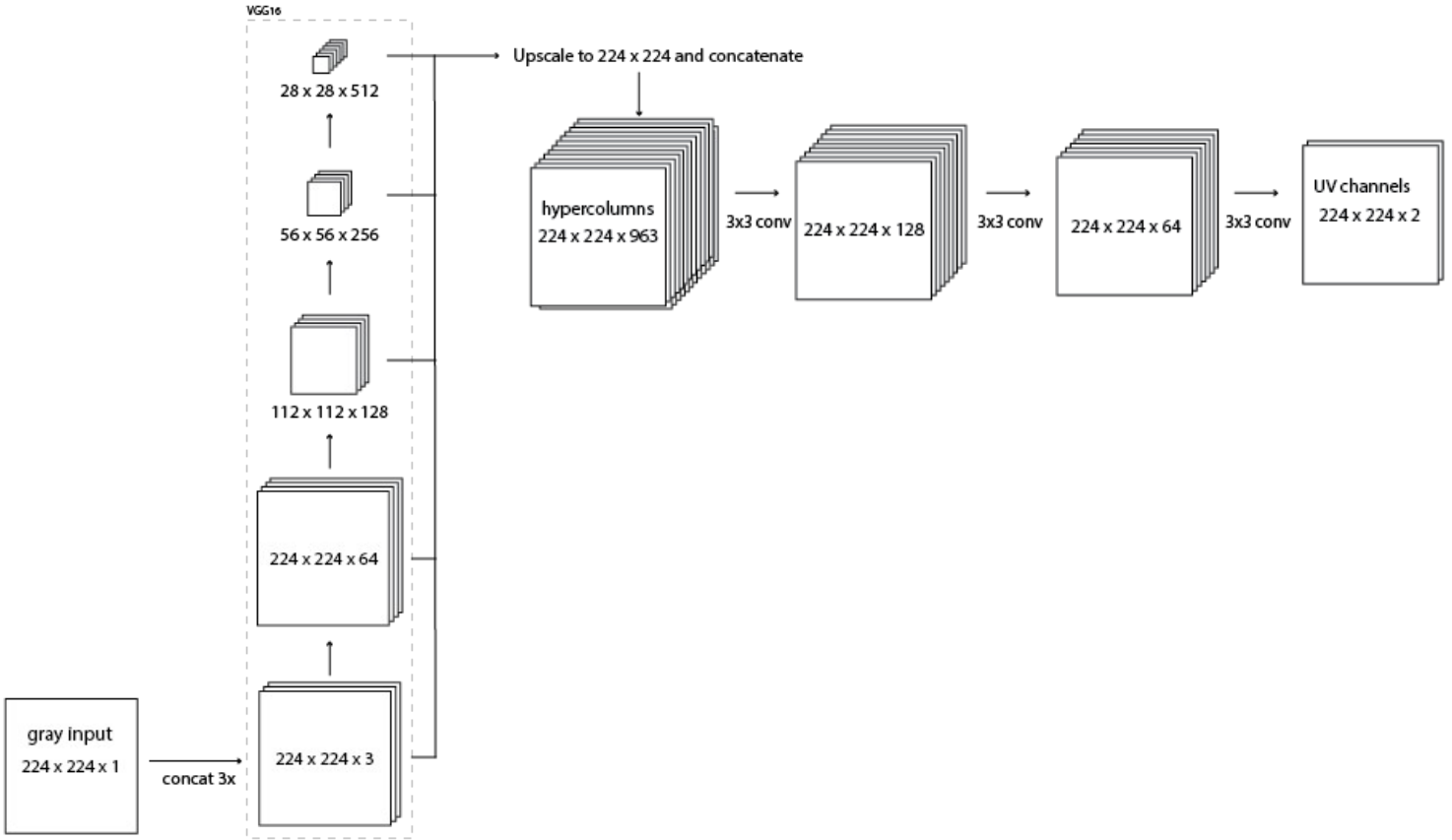


Dataset

http://tinyclouds.org/colorize

Image coloring: hypercolumns

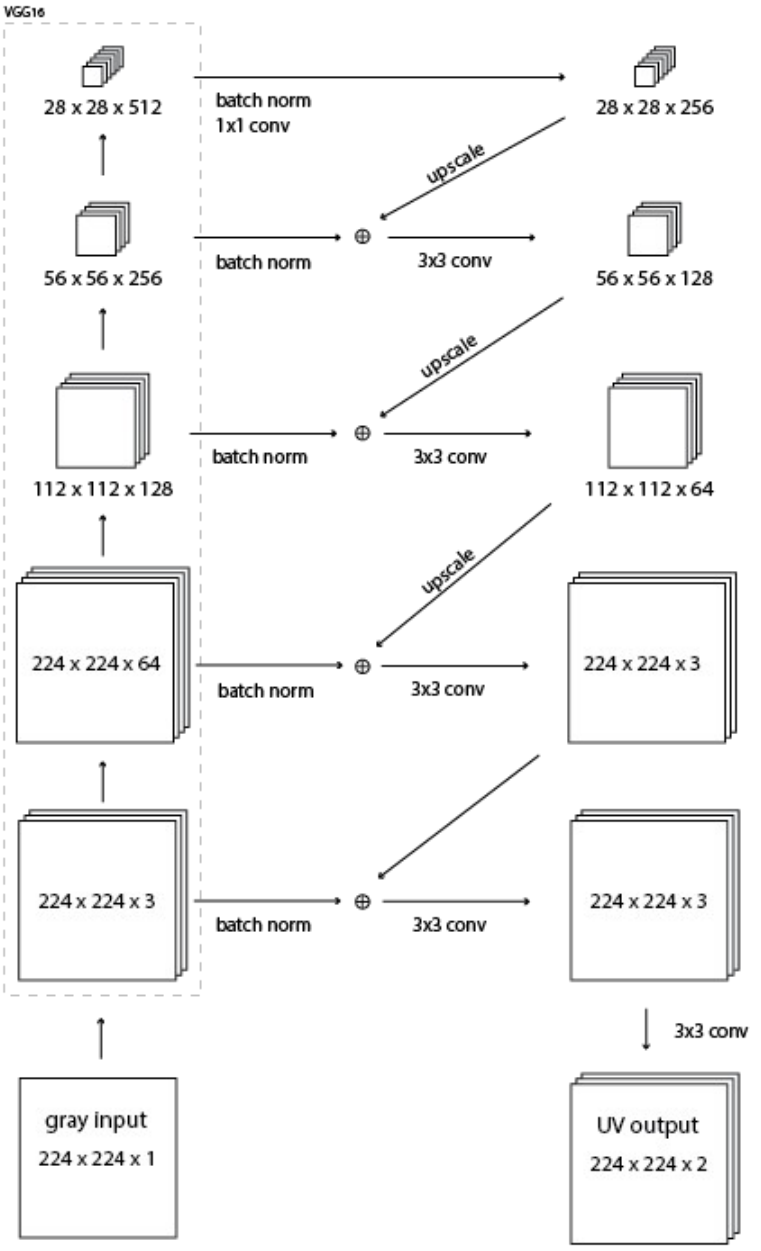
- Idea: upscale all layers to image size and pixel-wise
- Predict 2 UV channels
- Grayscale & UV → RGB



http://tinyclouds.org/colorize

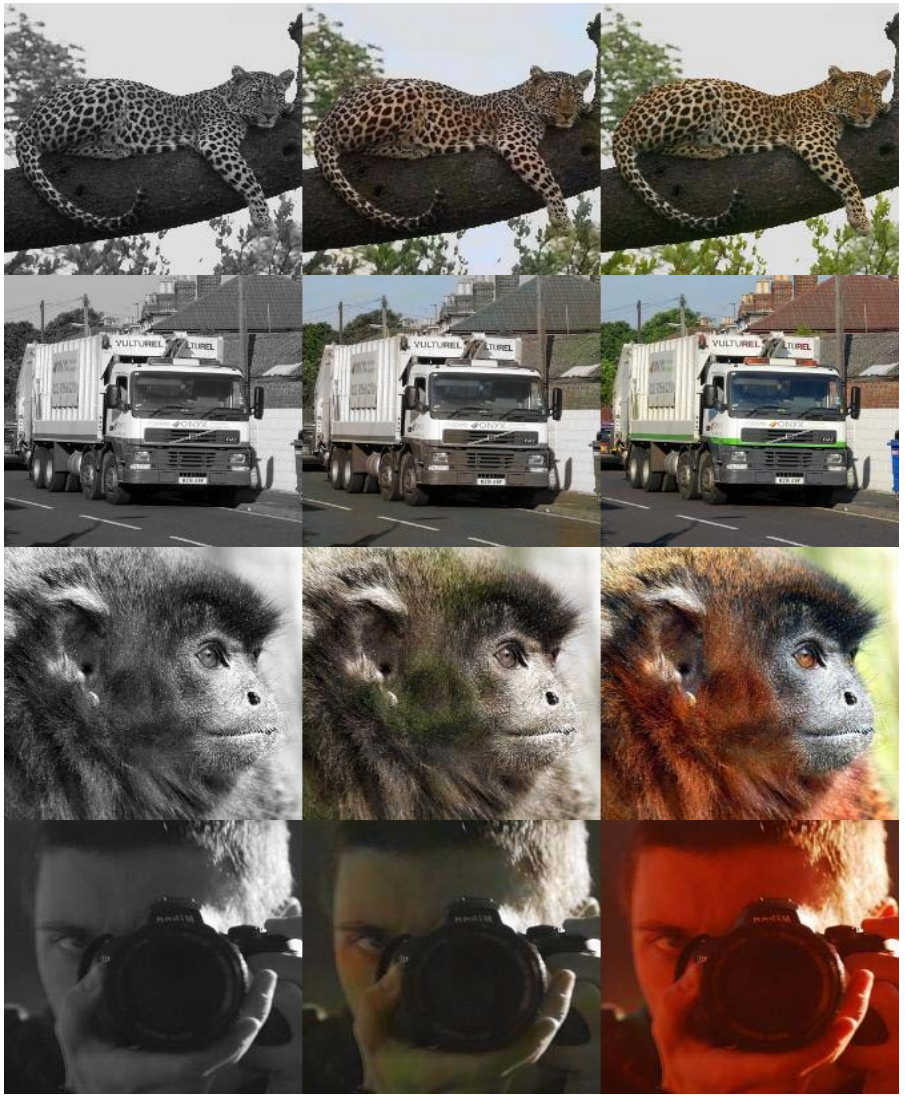
Image coloring: residual

- Similar to U-net
- Add instead of concat
  - Like ResNet
- Consumes less memory
  - No W x H x 1k blob



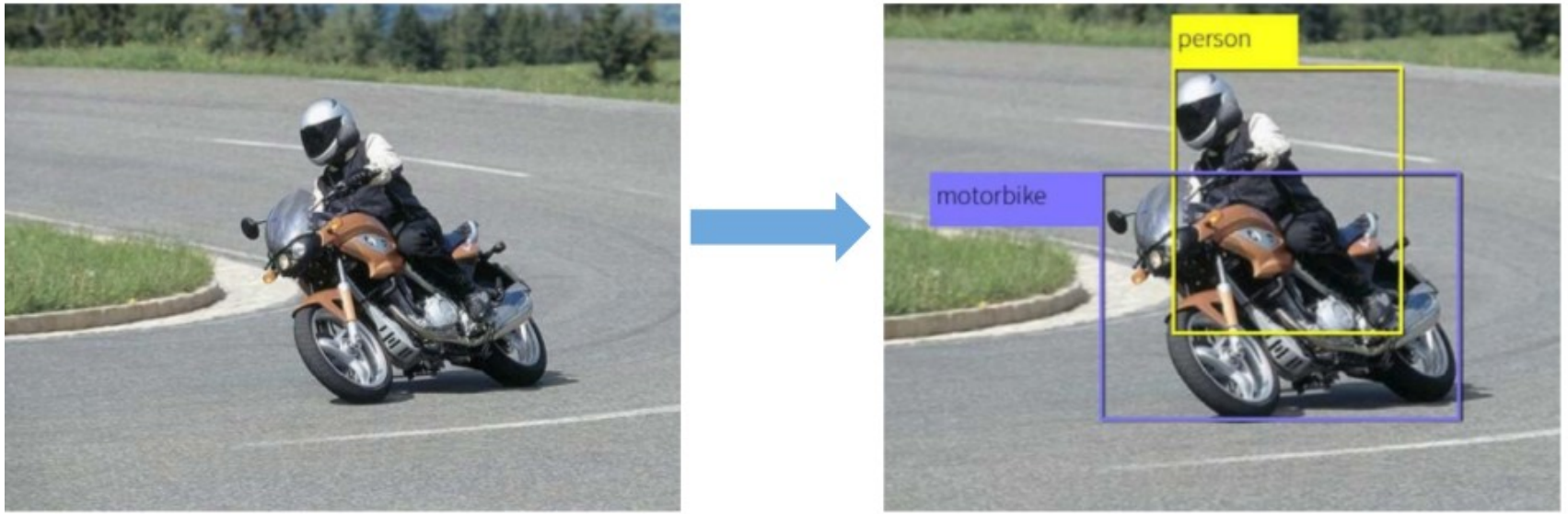
http://tinyclouds.org/colorize

Image coloring: examples



Input/prediction/reference

Bounding Box Regression



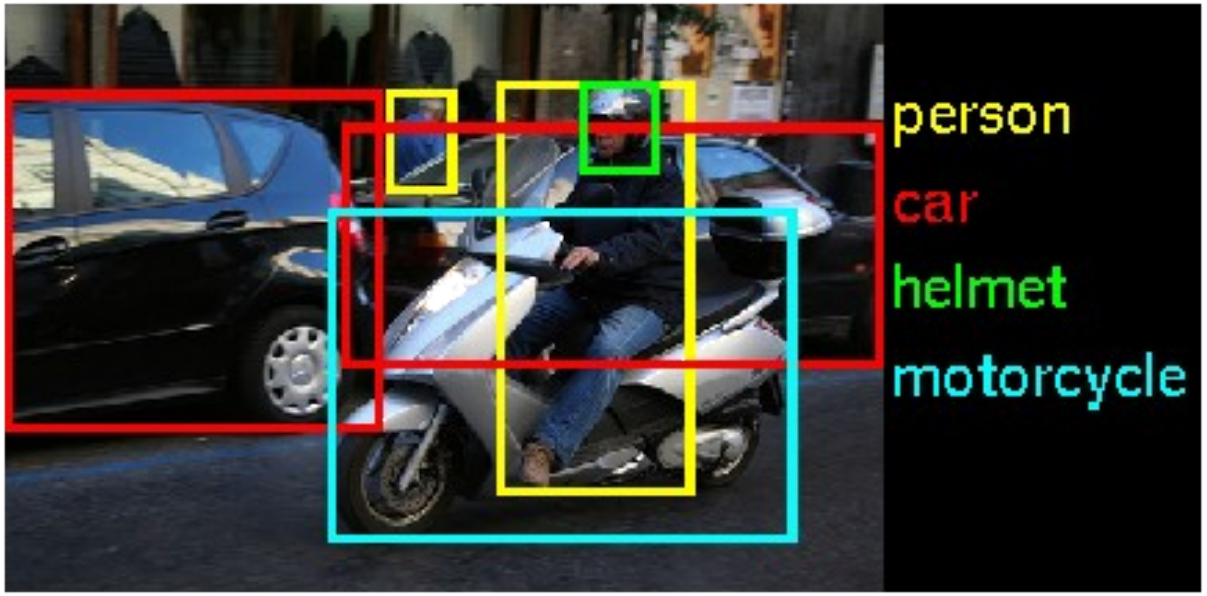
Input image

Desired output

- Predict bounding rectangle for the object
- How do we deal with a single square object out of 100 possible objects?

Ideas?

Bounding Box Regression



- Multiple squares?
- Non-square shape?
- Arbitrary rotation?

RCNN

- Generate candidates
- Adjust region shapes
- Apply classifier to each region

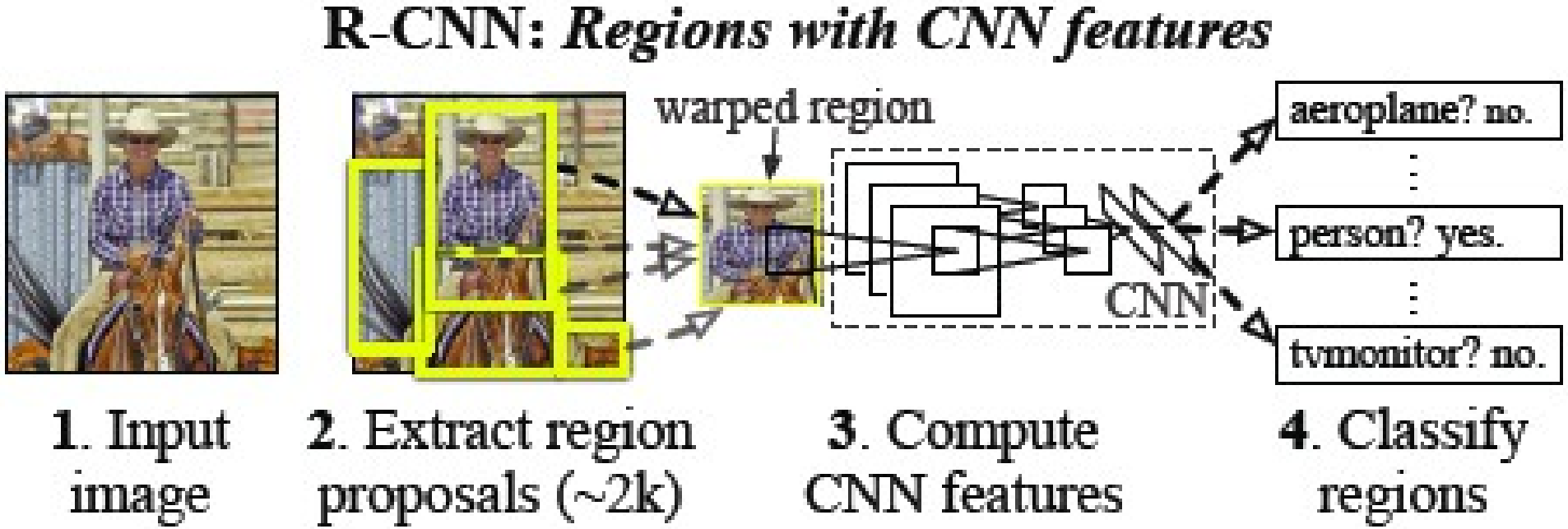
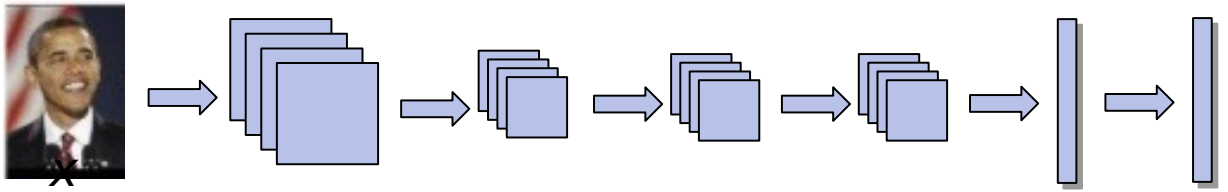


Image verification

- Image2vec



$$f_\theta(x) \in \mathbb{R}^n$$

- **We want:**

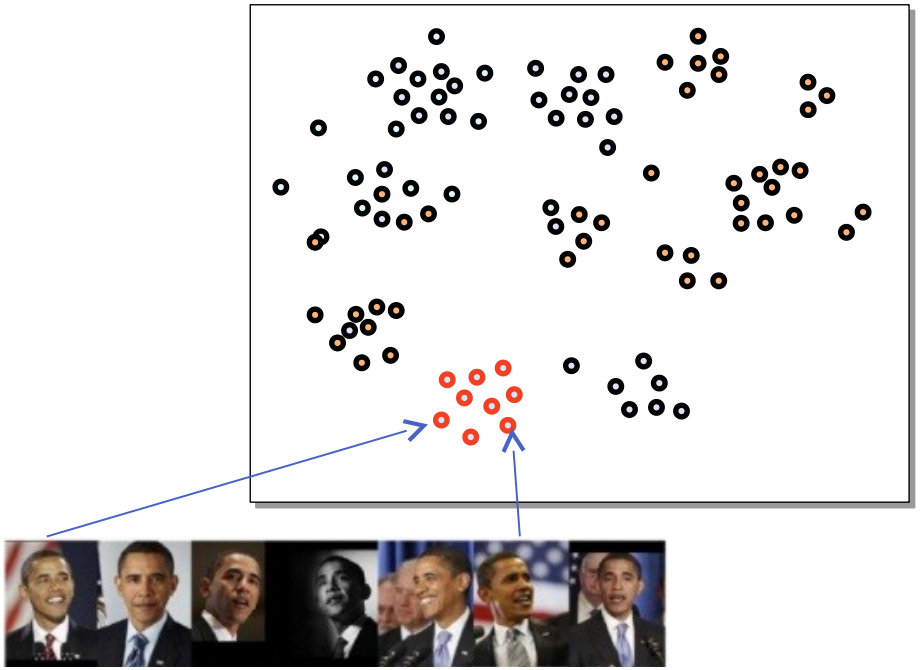
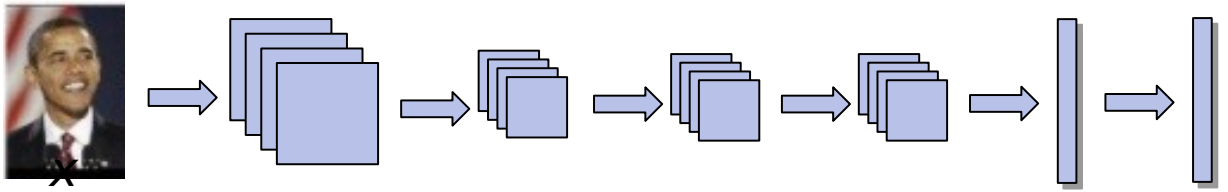


Image verification

- Image2vec



$$f_\theta(x) \in \mathbb{R}^n$$

- Similarity metric(example)

$$M(x1, x2) = \cos\left(f_\theta(x1), f_\theta(x2)\right)$$

- x1, x2 - pair of images, T = #same person

Loss:

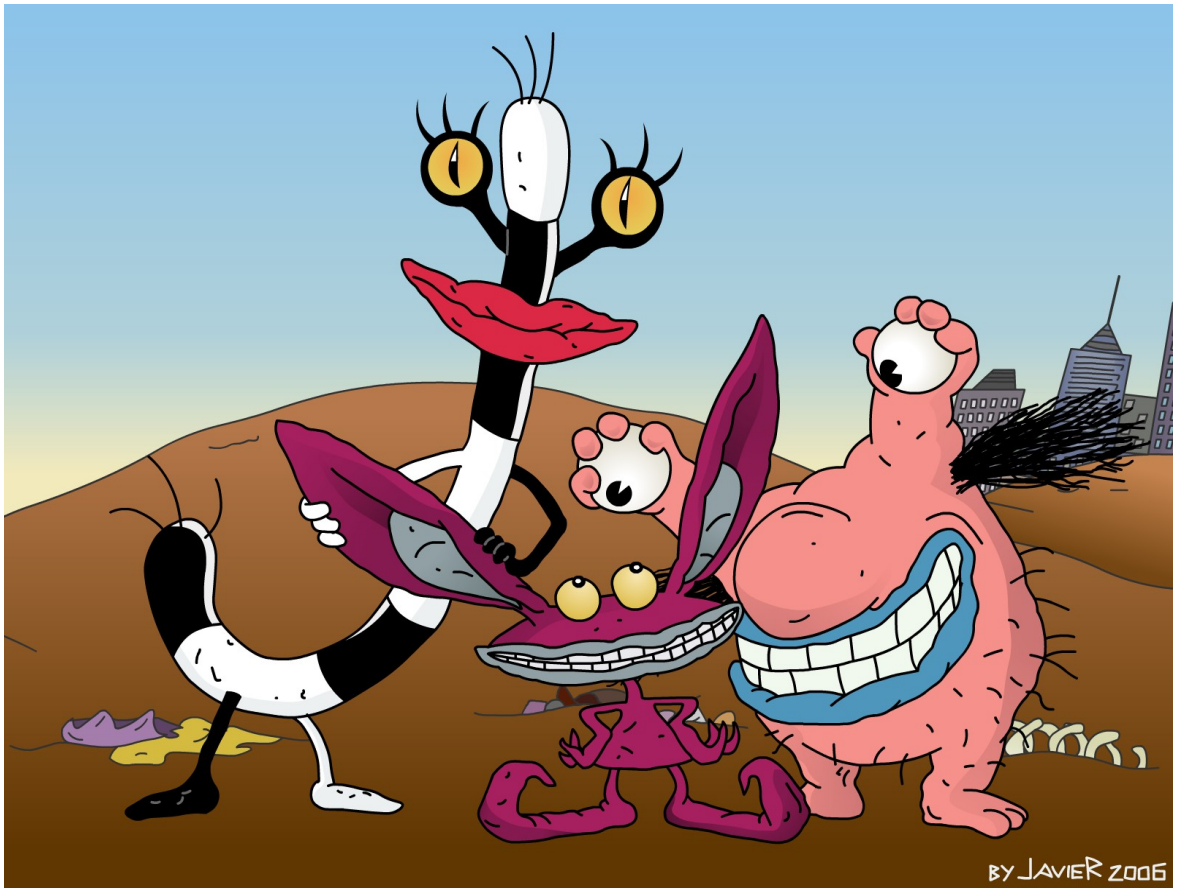
$$L = T \cdot M(x1, x2) + [1 - T] \cdot [1 - M(x1, x2)]$$

Speed up:

- Precompute all query vectors O(n queries)
- Precompute all image vectors O(n images)
- Compute cosine in prediction time
- Locally Sensitive Hashing

...

Google Yandex



Model zoo in today's seminar