Demystifying Deep Learning:
An Introduction to convolutional neural networks and computer vision

Justin Ellis

RTA Meetup
February 20, 2018

Talk Overview

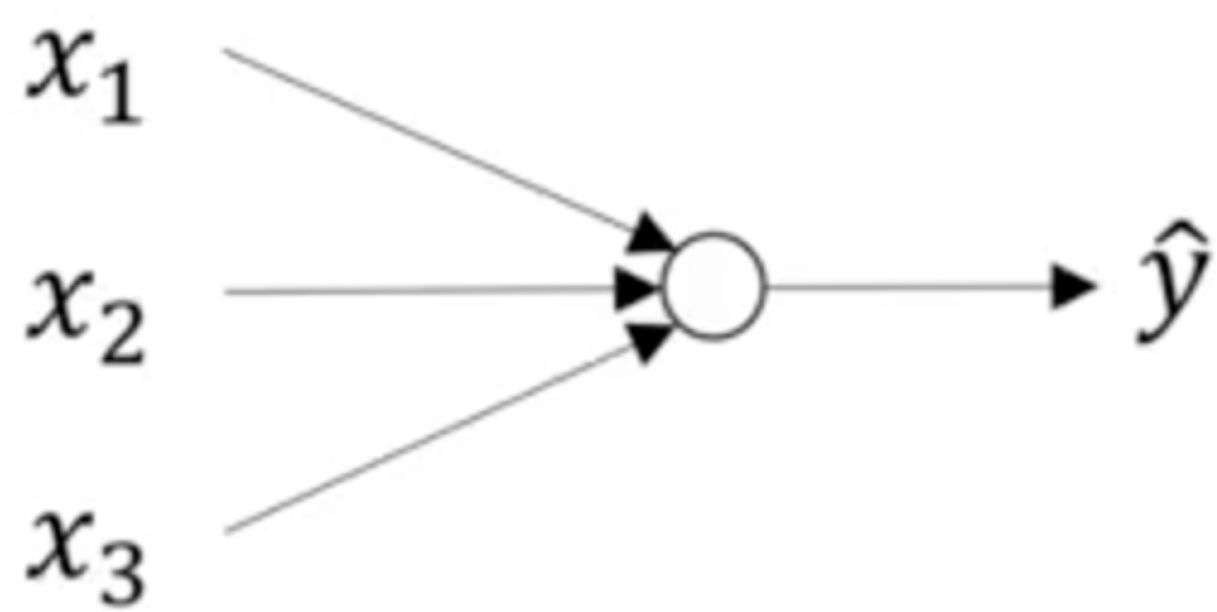
- Overview of Deep learning
- Convolutional Neural Networks (CNN)
- Applications of computer vision
- Transfer learning

Myths

- Neural networks are a black box.
- You needs tons of data and a cluster of GPUs to use deep learning
- You need years of training and tons of math to use and understand deep learning.

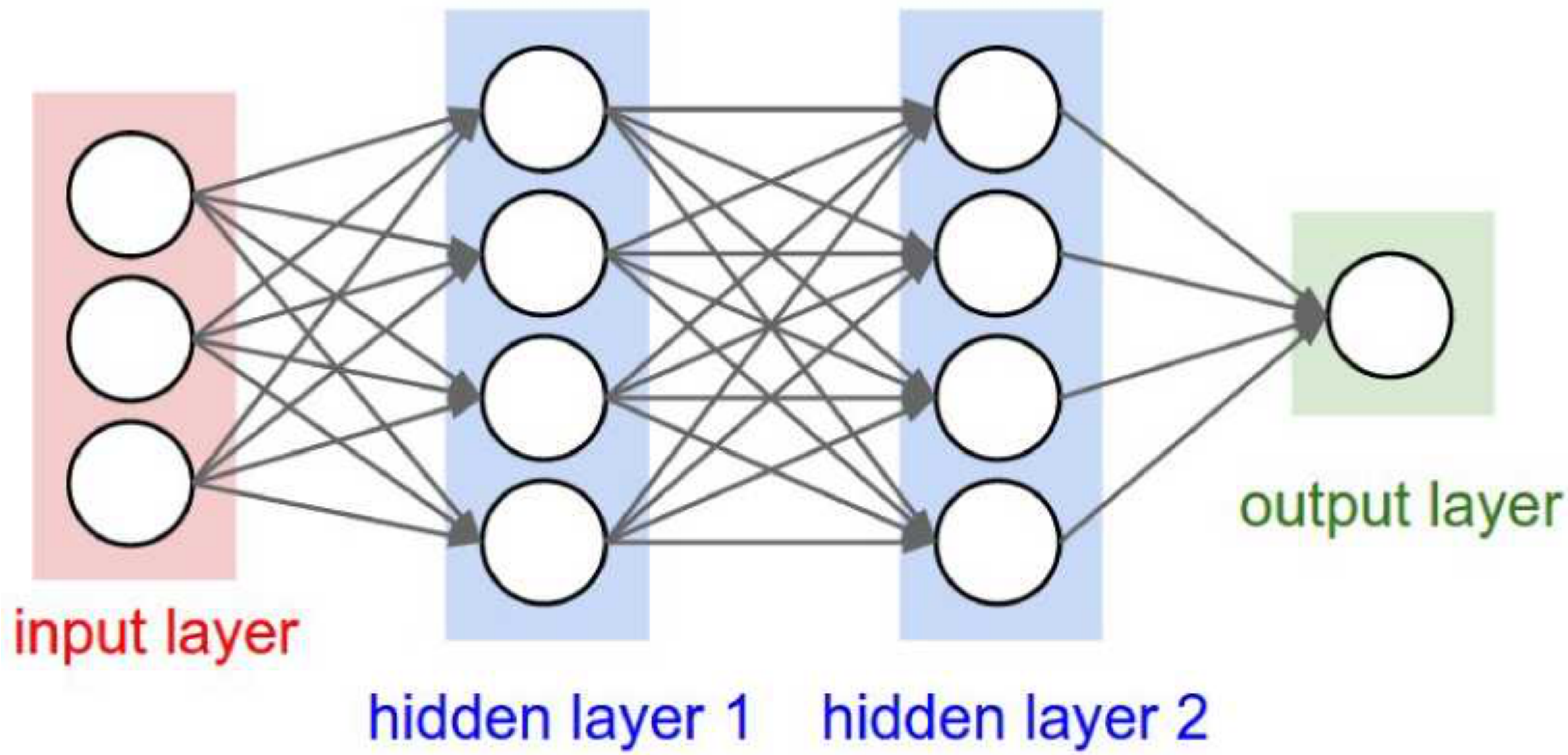
A familiar algorithm

- Logistic regression
  - Prediction $\hat{y} = \mathrm{Sigmoid}(Wx + b)$
  - Activation function $\mathrm{Sigmoid}(z) = \dfrac{1}{1 + e^{-z}}$
  - Cost function $J(W, b) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$
- Can be represented with

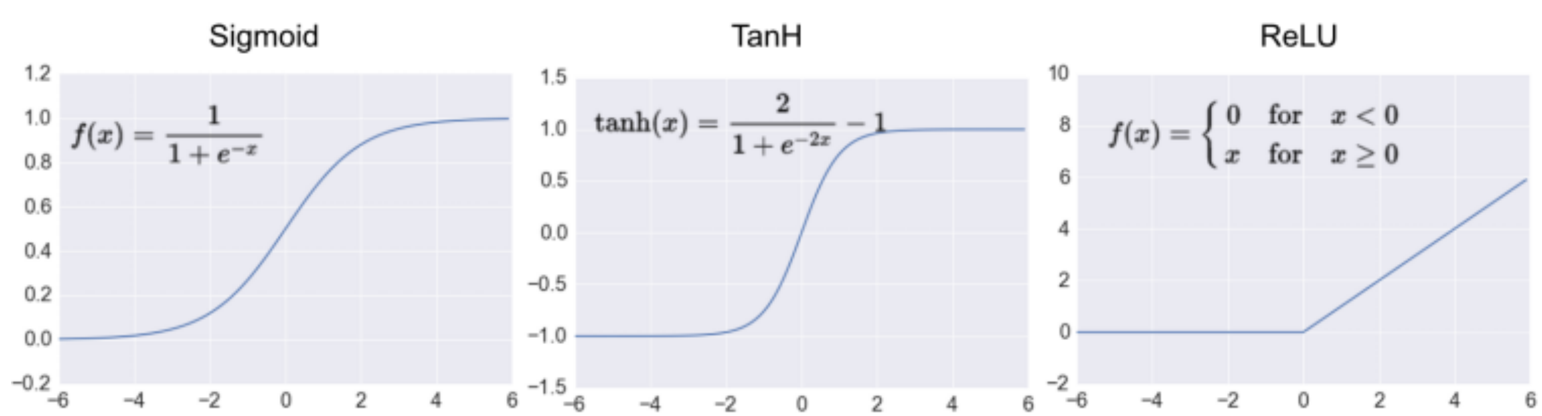


Deep Neural Networks

$$x = a^{[0]} \qquad a^{[\ell]} = g^{[\ell]}(W^{[\ell]}a^{[\ell-1]} + b^{[\ell]})$$



$$g(z) =$$



Deep Neural Networks Simple Keras Code

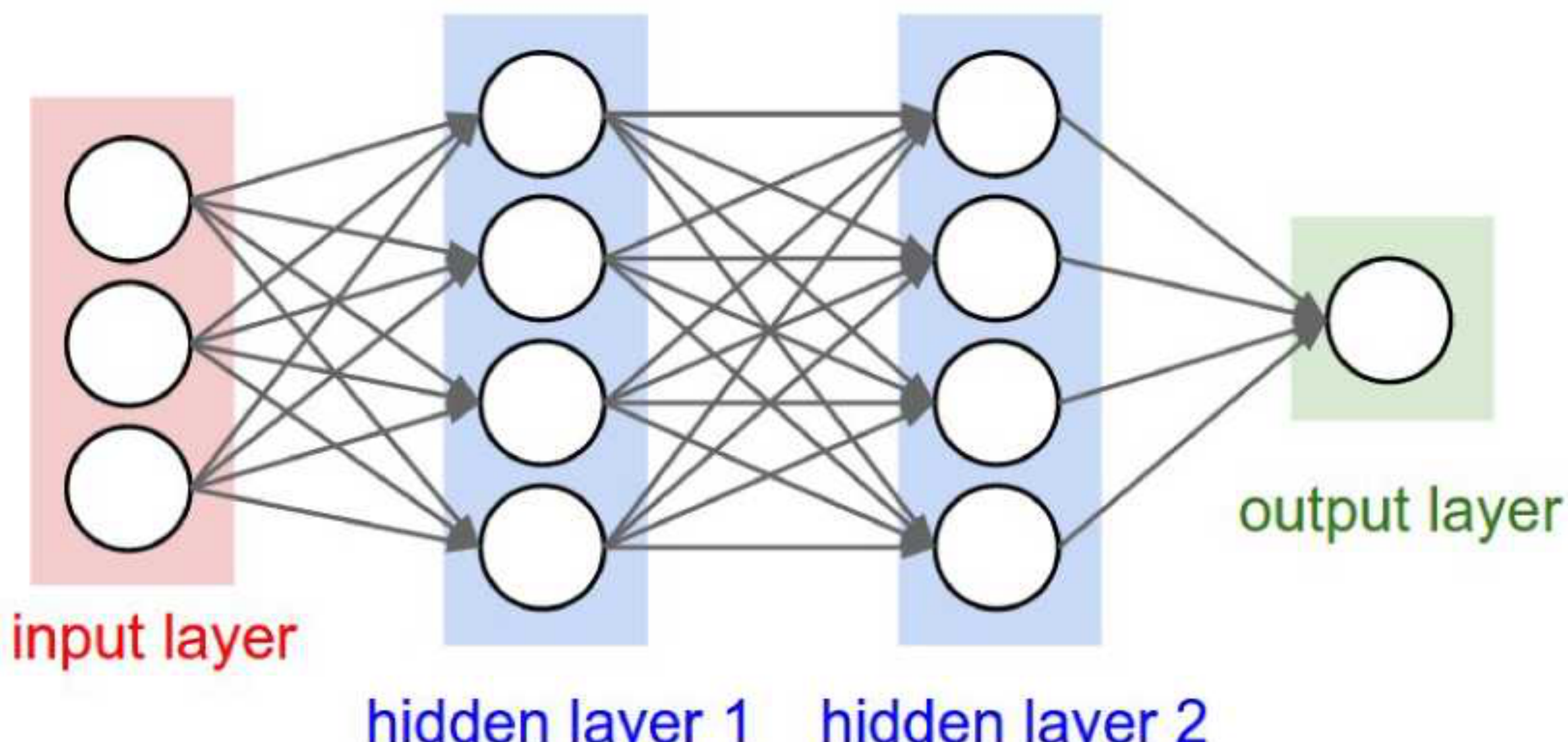
- With modern DL frameworks, most neural nets can be coded in <~ 100 lines

```
# create model
model = Sequential()
layer_sizes = [64, 128, 512, 512, 128, 64]
model.add(Dense(layer_sizes[0], input_dim=64, activation='relu'))
for ls in layer_sizes[1:]:
    model.add(Dense(ls, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fit the model
model.fit(X, Y)
```

- Will work on CPU or GPU with no change to code!

How could we use this for computer vision?

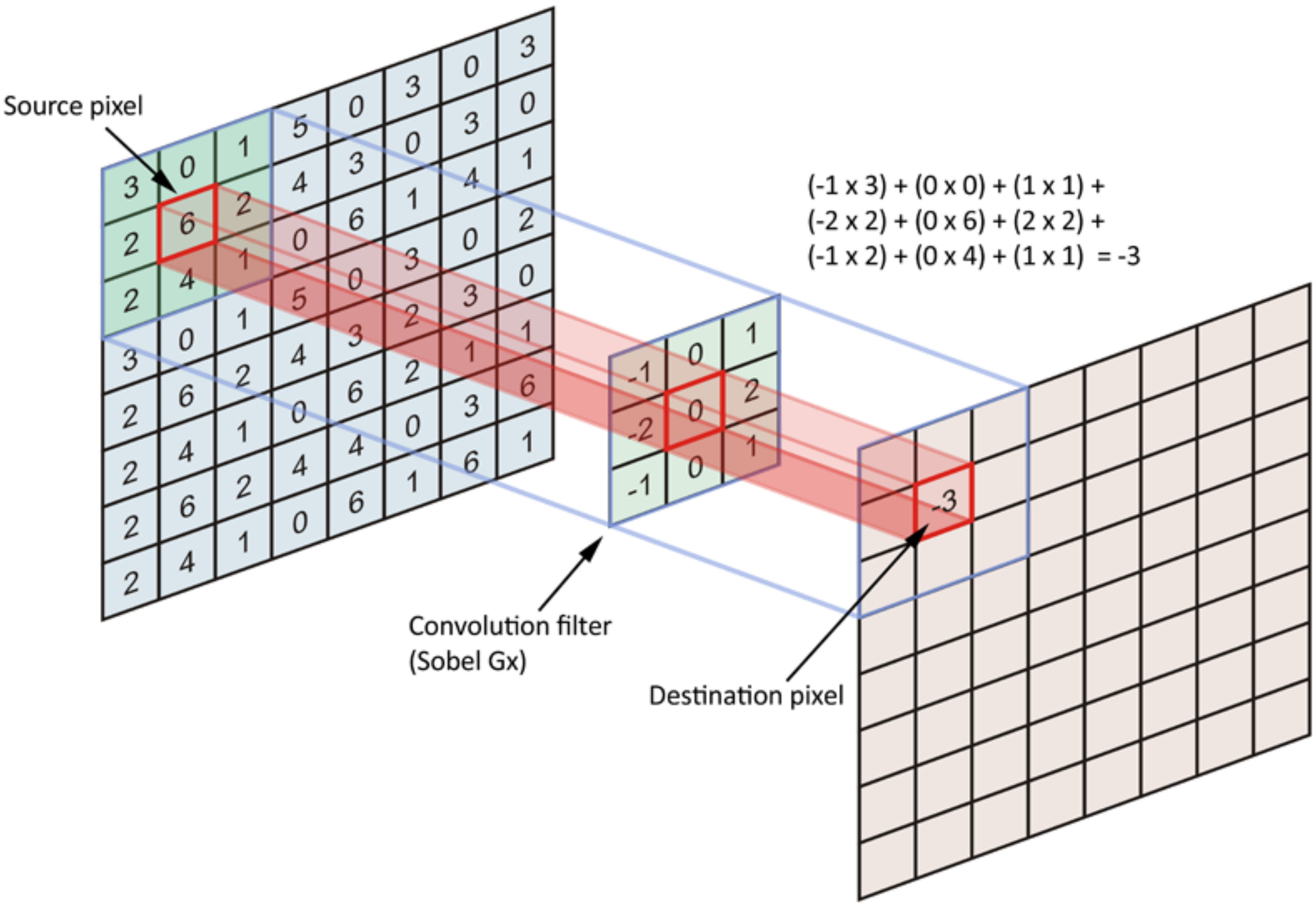
- Say image is 224 x 224 x 3 (RGB).
- Unroll to feature vector of size 150528
- For just the first hidden layer with 100 nodes we have over 15 million parameters. Not good

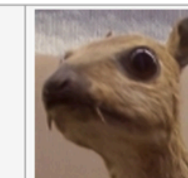
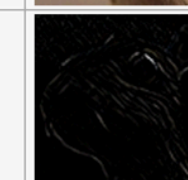
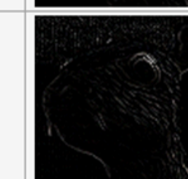
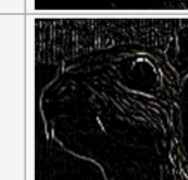
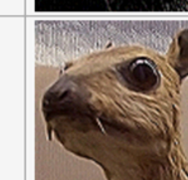
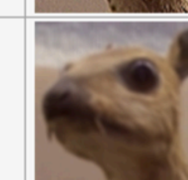
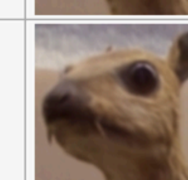


Images

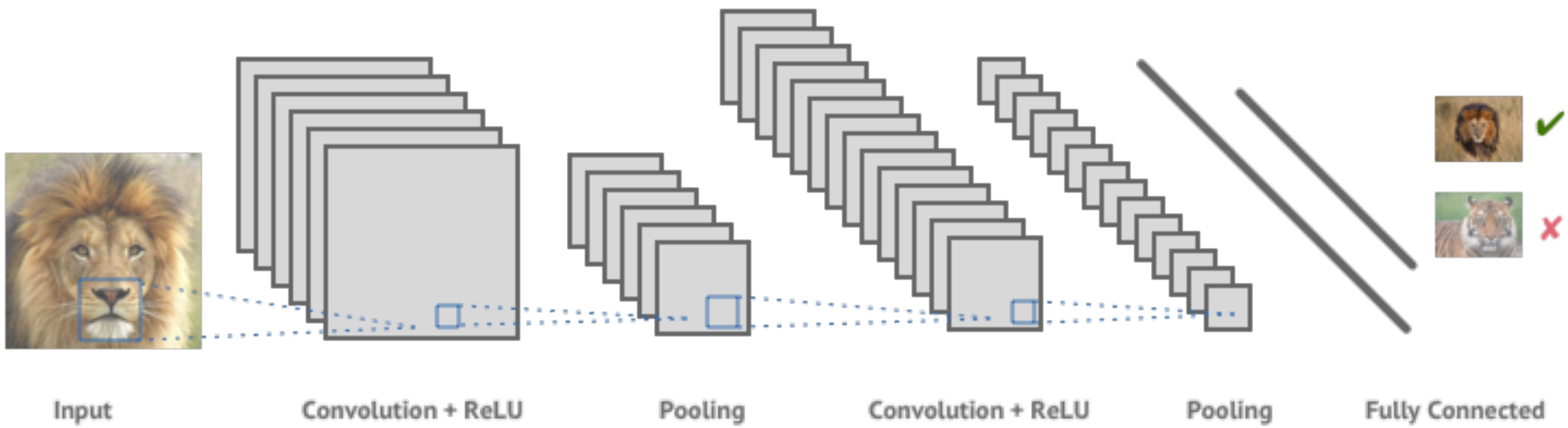
- What do you see?
- Blob in middle?
- Black, white, red?
- Animal?
- Dog?
- Poodle?
- Happy?

Convolutions



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Basic Convolutional Neural Network (CNN) Architecture

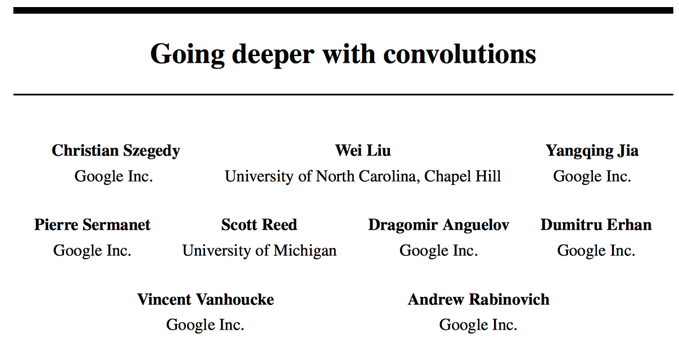


Source: https://shafeentejani.github.io/2016-12-20/convolutional-neural-nets/

- Uses filters to *learn* features of images (i.e. edges, colors, shapes, etc)
- Standard architecture has many layers of the form CONV->POOL followed by some fully connected layers at the end where the last layer depends on the task (sigmoid, softmax, etc)
- Different architectures use different filter sizes, different padding, different depths, etc

CNN Architecture

ZOO



Going deeper with convolutions

Christian Szegedy
Google Inc.

Wei Liu
University of North Carolina, Chapel Hill

Yangqing Jia
Google Inc.

Pierre Sermanet
Google Inc.

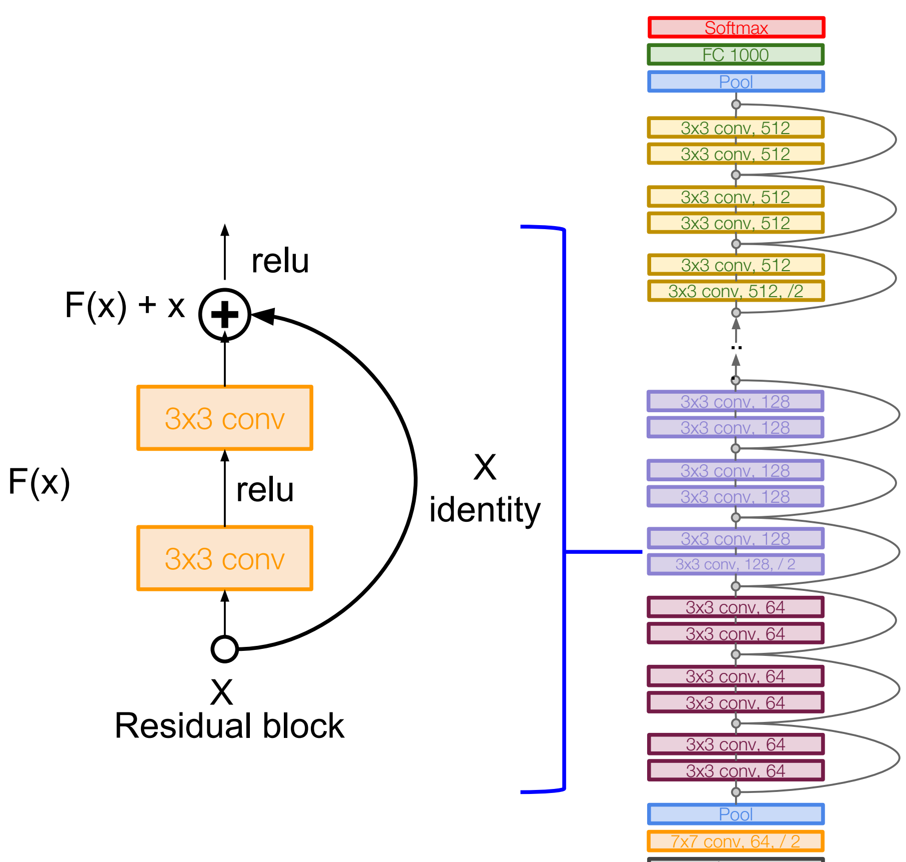
Scott Reed
University of Michigan

Dragomir Anguelov
Google Inc.

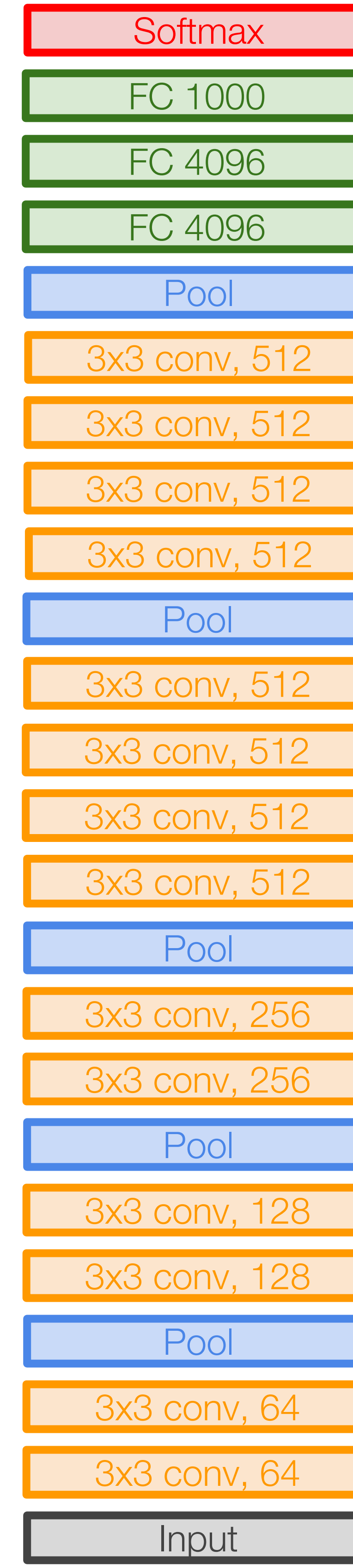
Dumitru Erhan
Google Inc.

Vincent Vanhoucke
Google Inc.

Andrew Rabinovich
Google Inc.

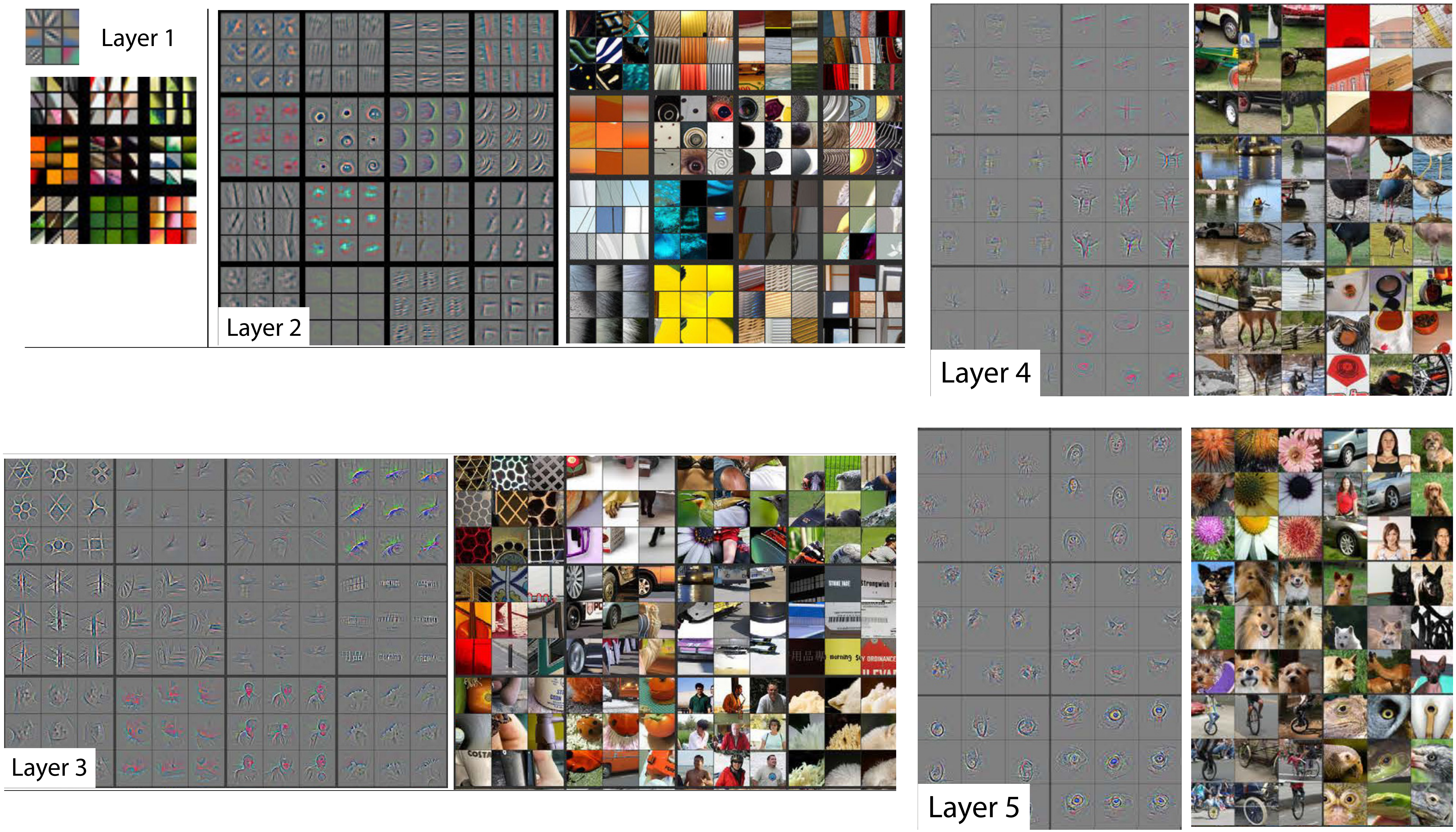


VGG16



VGG19

What does a CNN “see”?

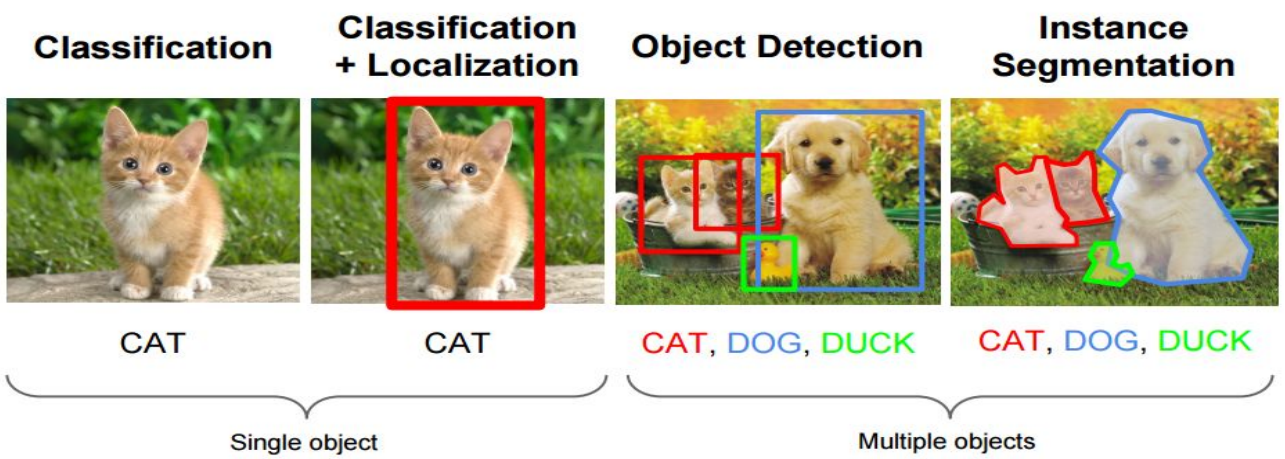


Zeiler & Fergus 2013, Visualizing and understanding convolutional networks

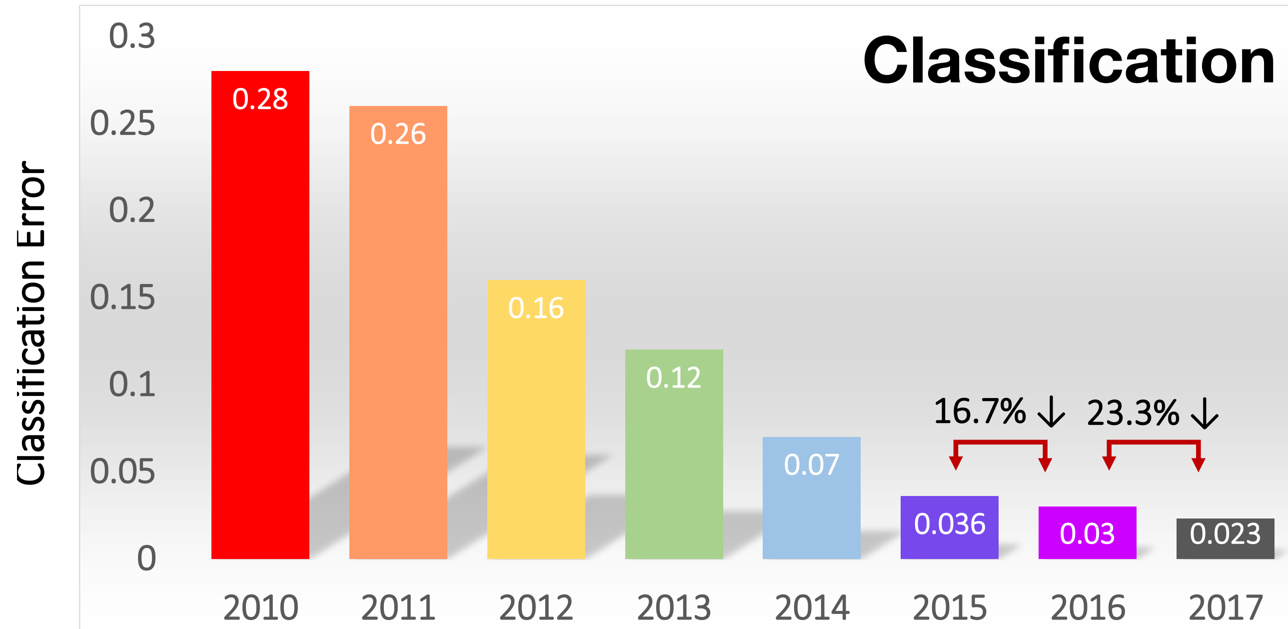
Computer Vision Tasks

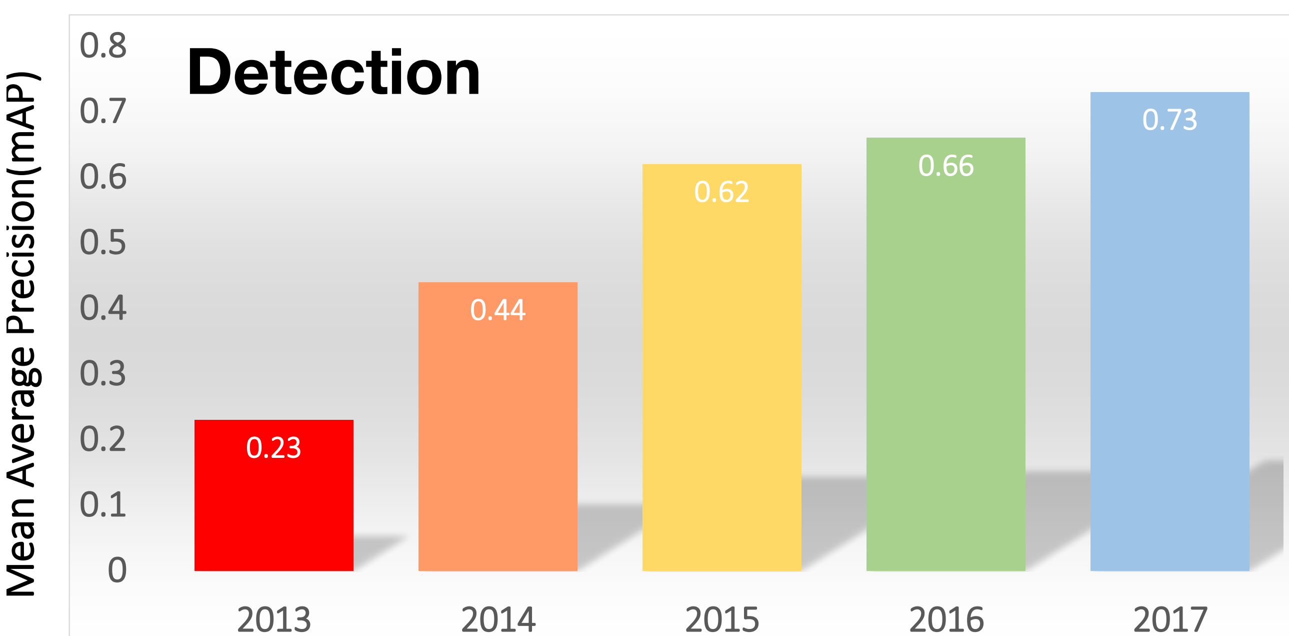
“the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images.”

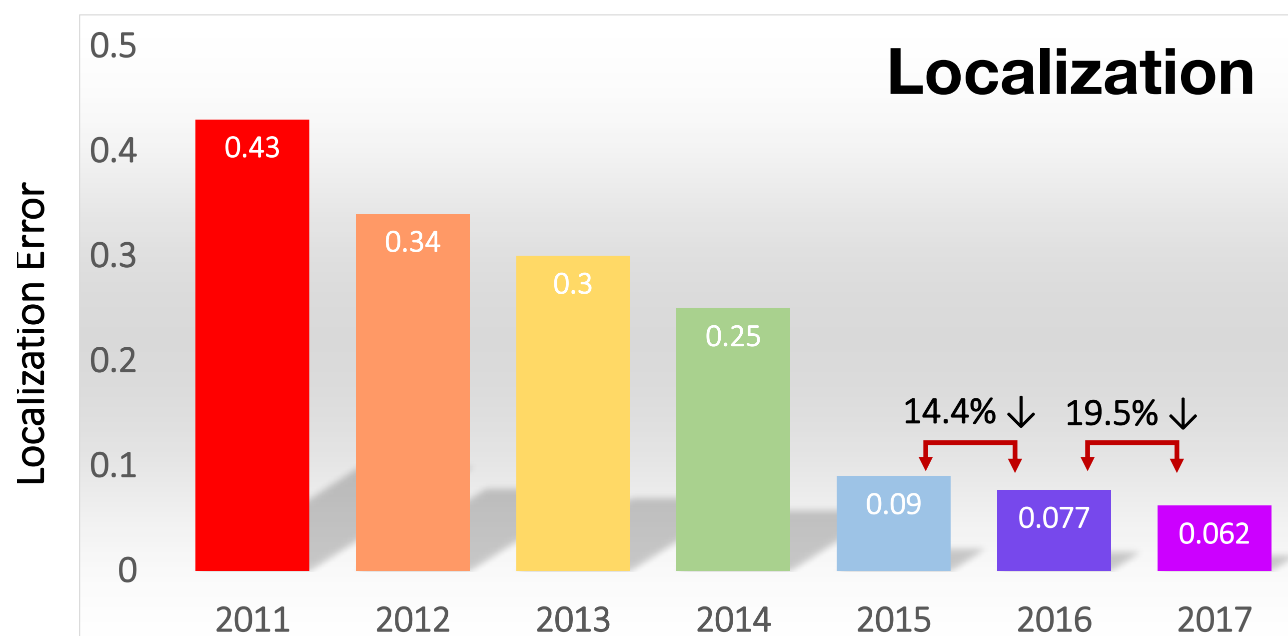
- British Machine Vision Association



Source: Fei-Fei Li, Andrej Karpathy & Justin Johnson (2016) cs231n, Lecture 8 - Slide 8, *Spatial Localization and Detection* (01/02/2016). Available: http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf



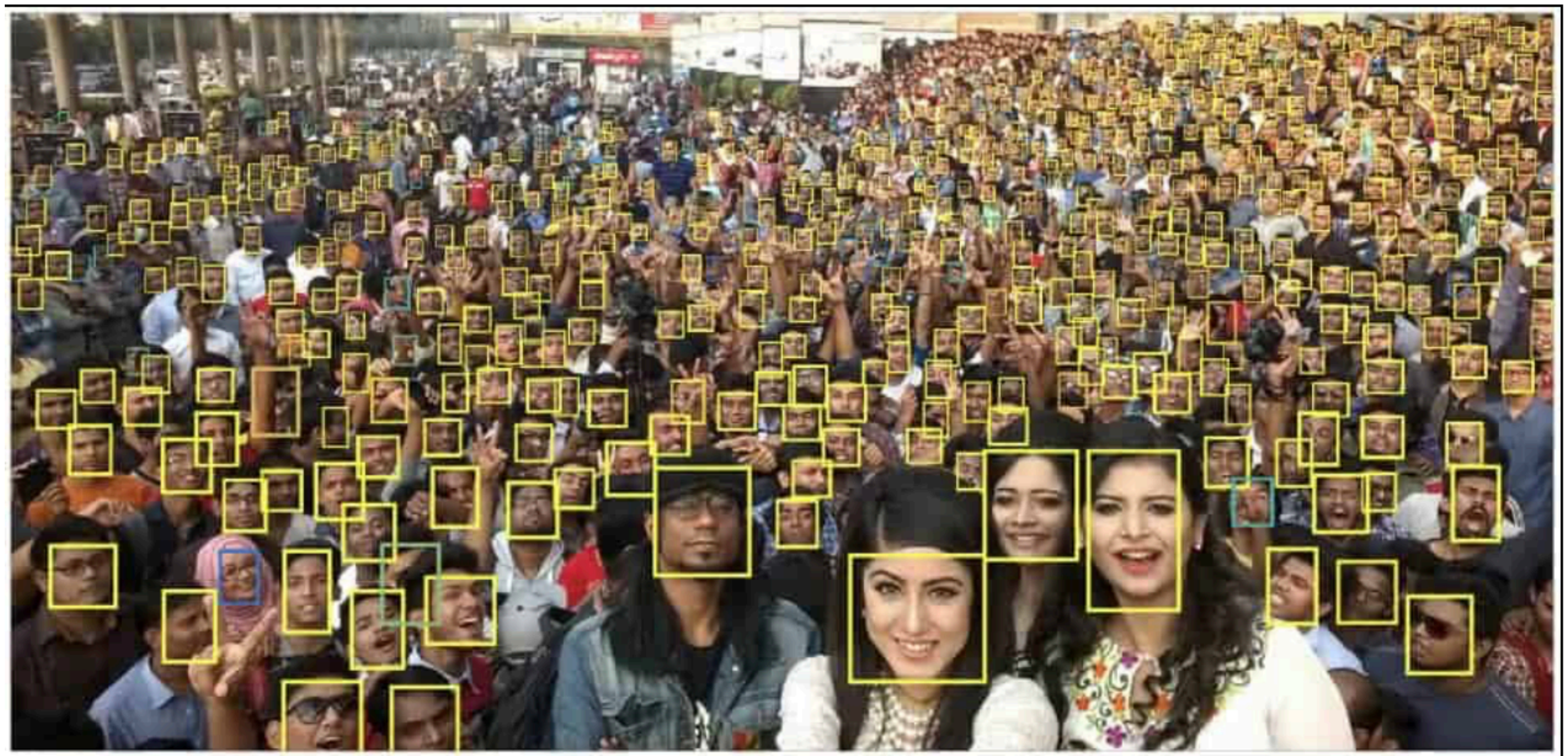


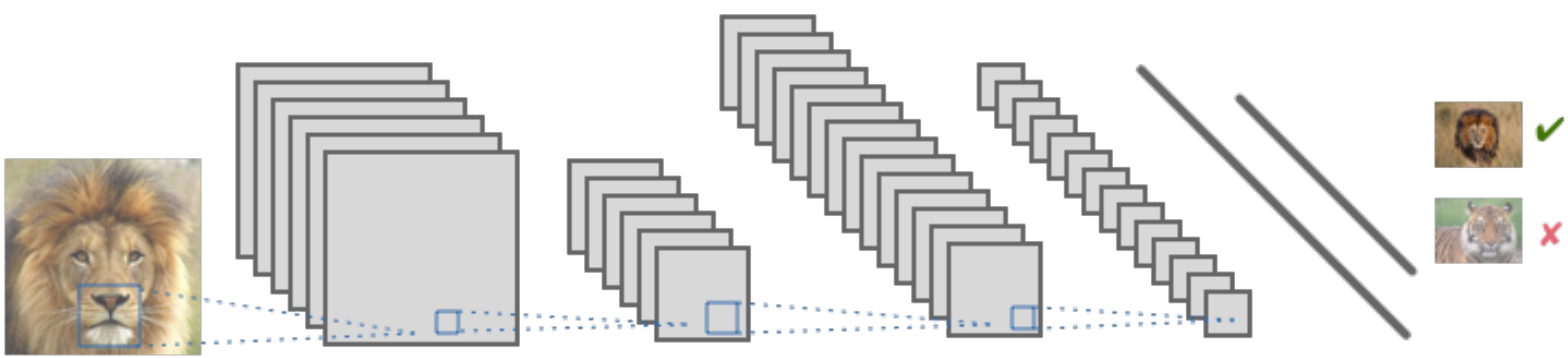


Note: precision = True Positives / Predicted Positives

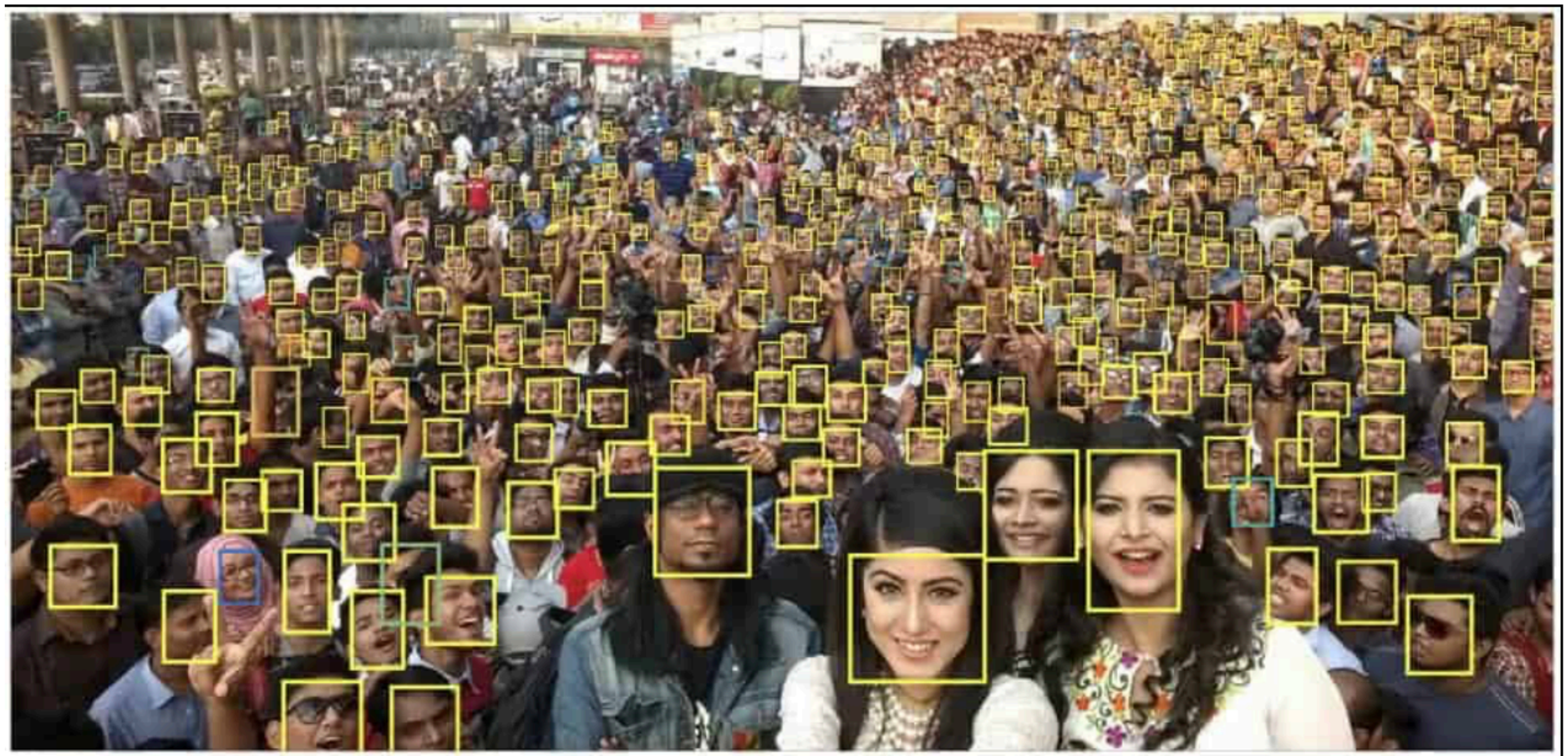
Source: http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf

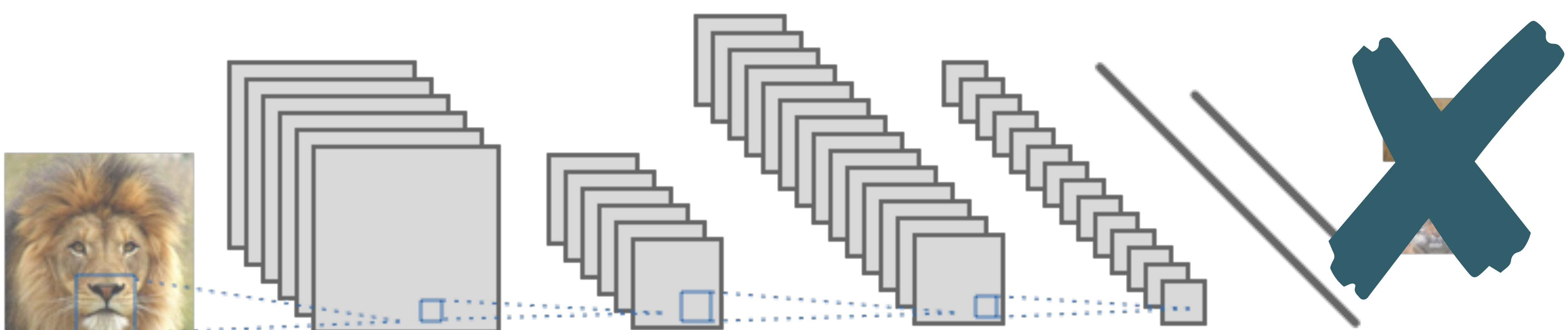
Facial Recognition





Facial Recognition





Neural Style Transfer

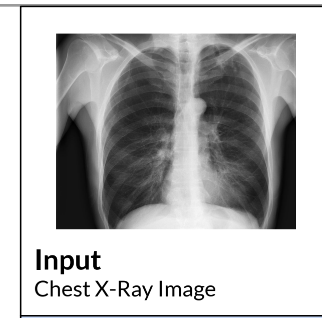
- Image generated by minimizing the difference between “style” and “content” image.

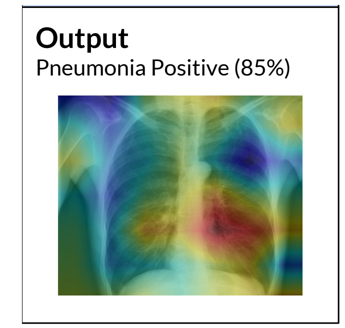
Real Time Object Detection

https://www.youtube.com/watch?v=VOC3huqHrss

(some) Computer Vision applications

- X-ray/MRI classification



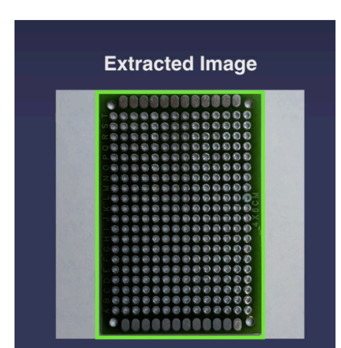


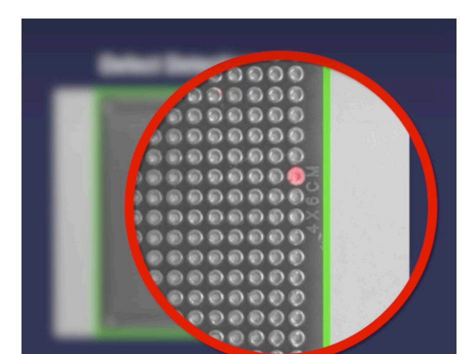
- Product discovery

- Insurance claims

$$ Payout

- Quality control

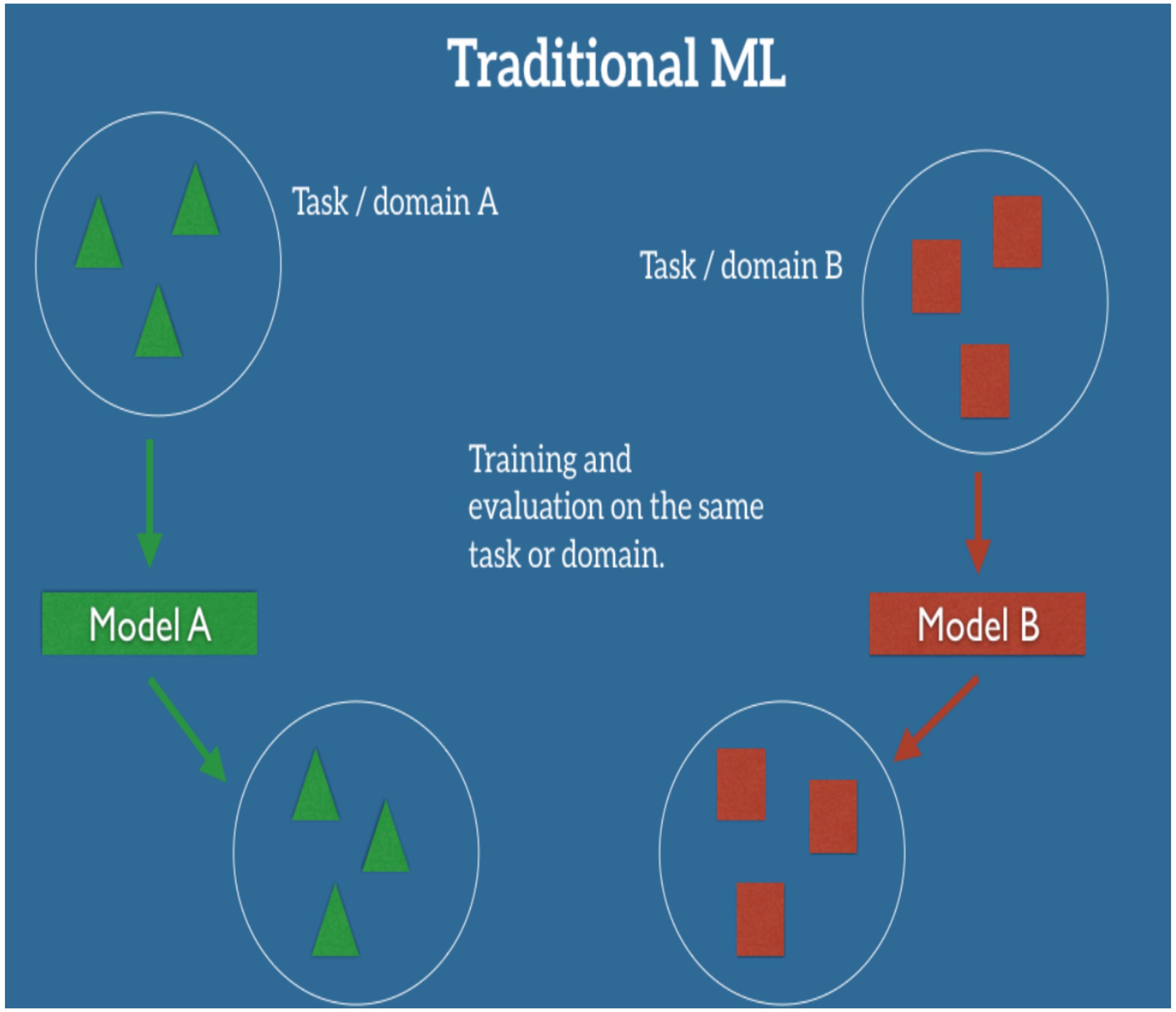


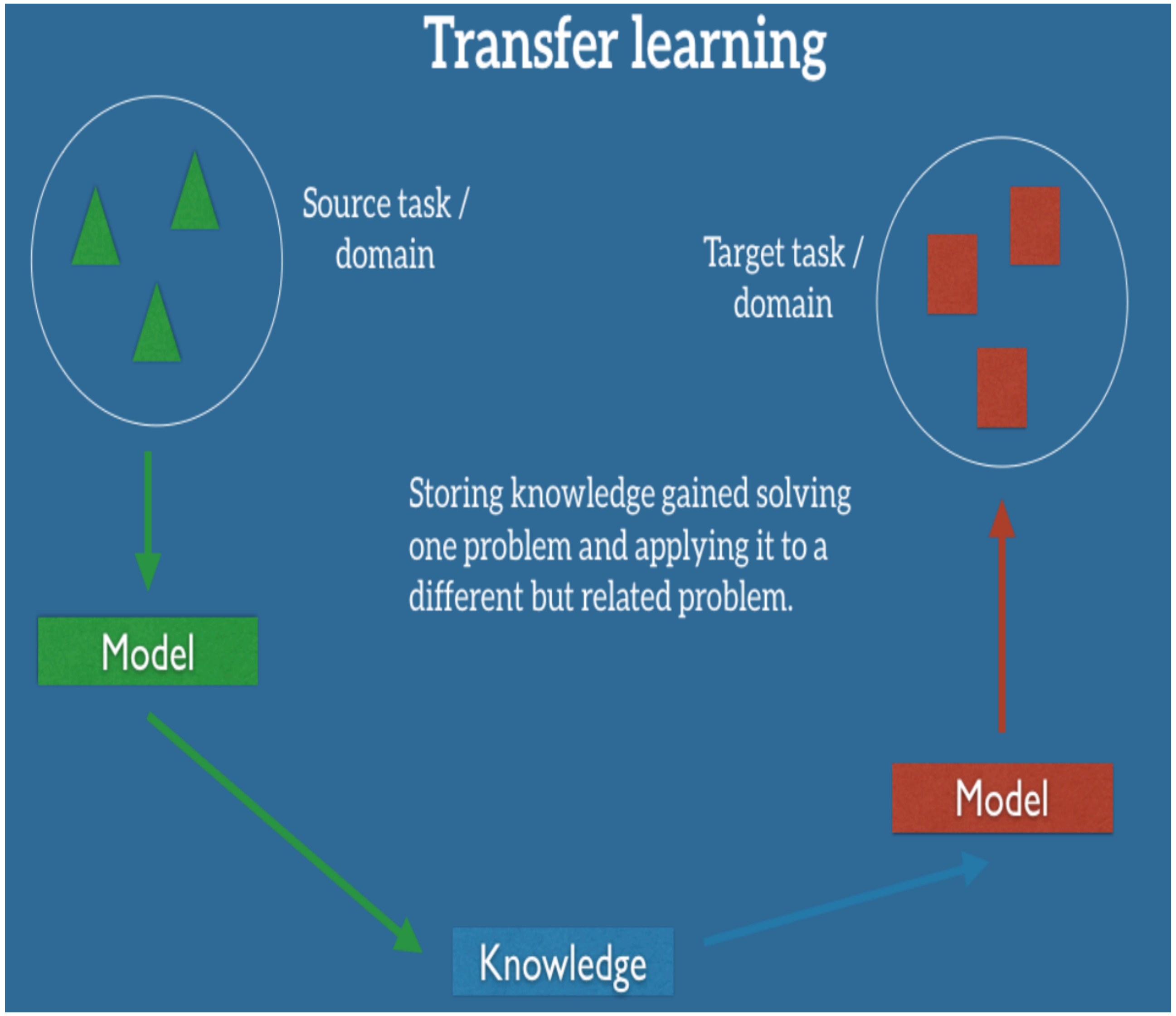


So how does one “do” deep learning

Transfer learning

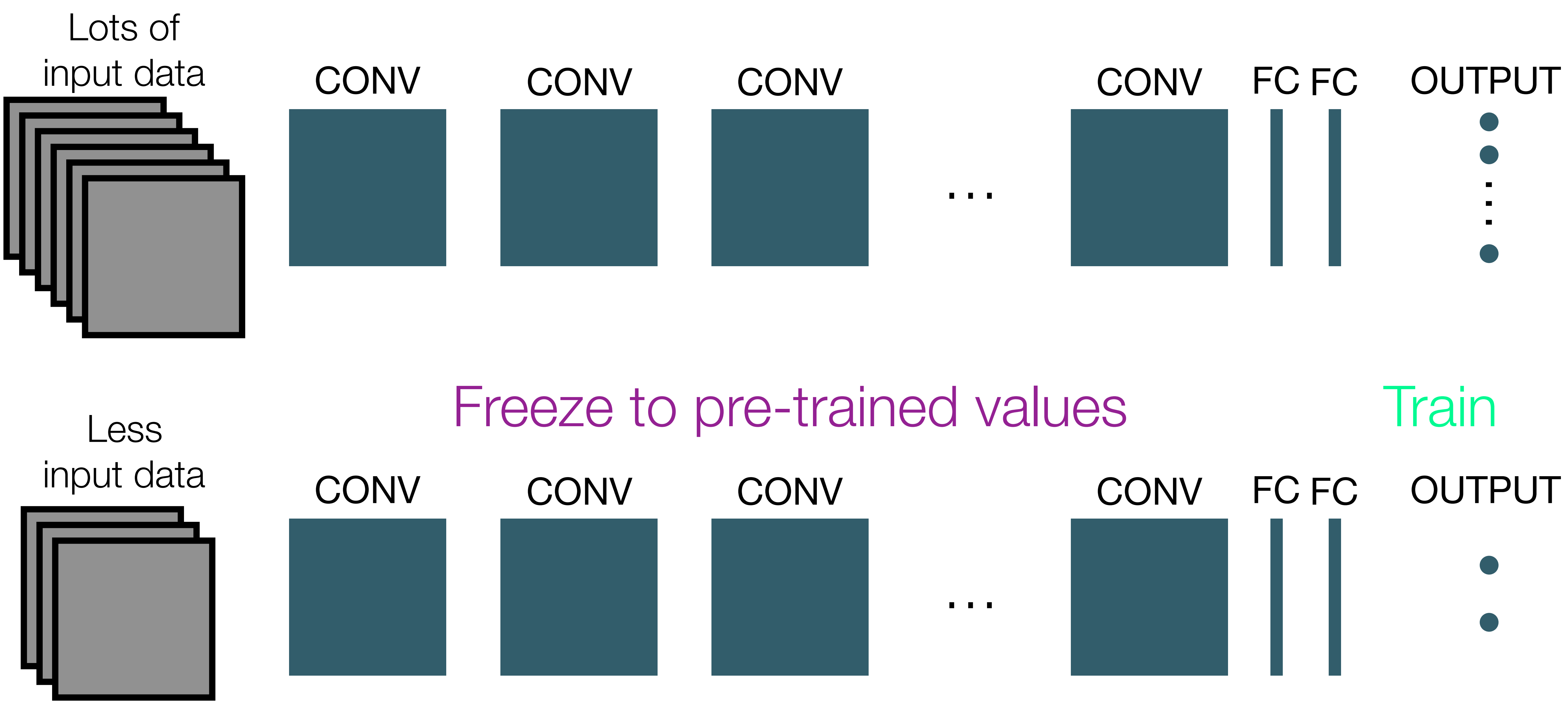
- Use “knowledge” from pre-trained network and “fine-tune” to new task





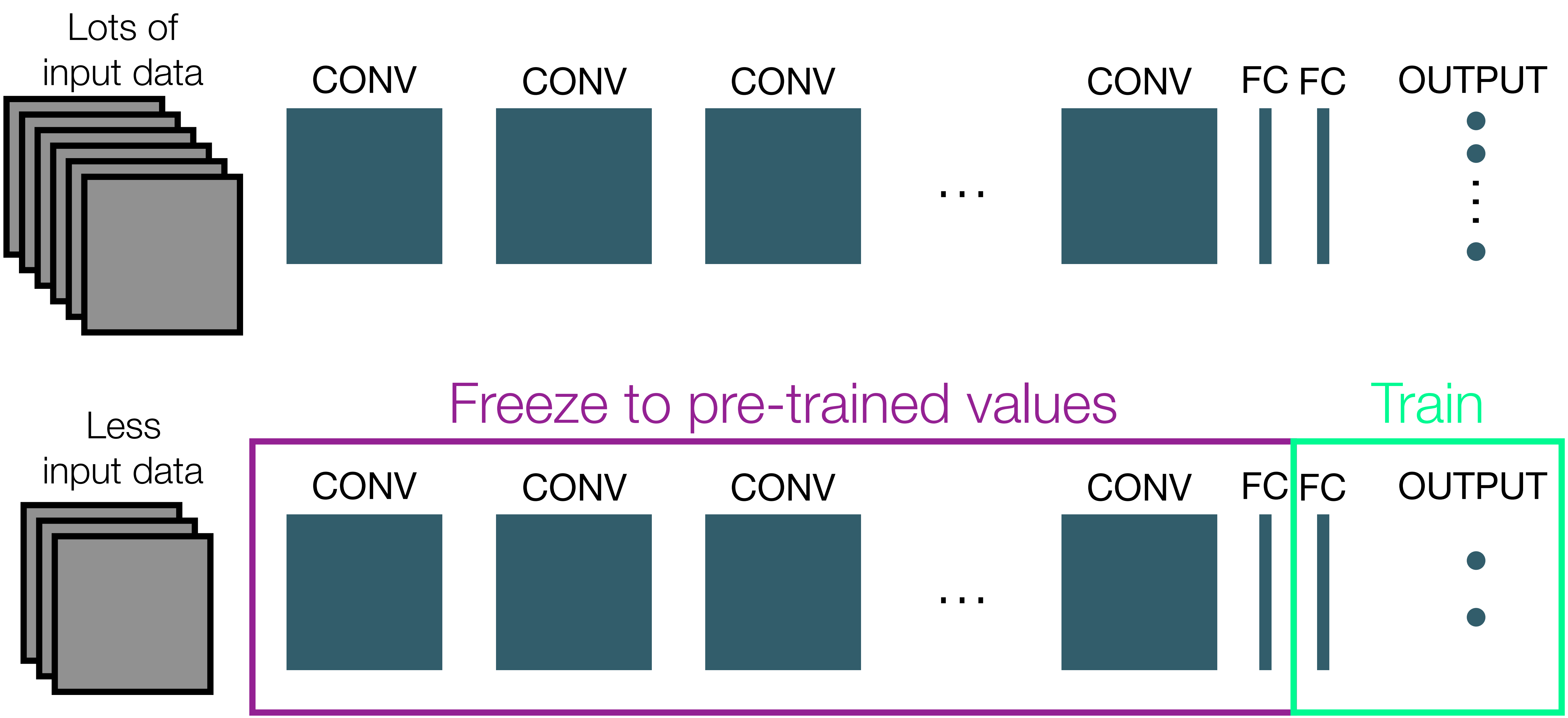
Transfer Learning Techniques for CV

- First train data on large available data set (i.e. imagnet) or better yet, download the pre-trained weights for a specific CNN architecture



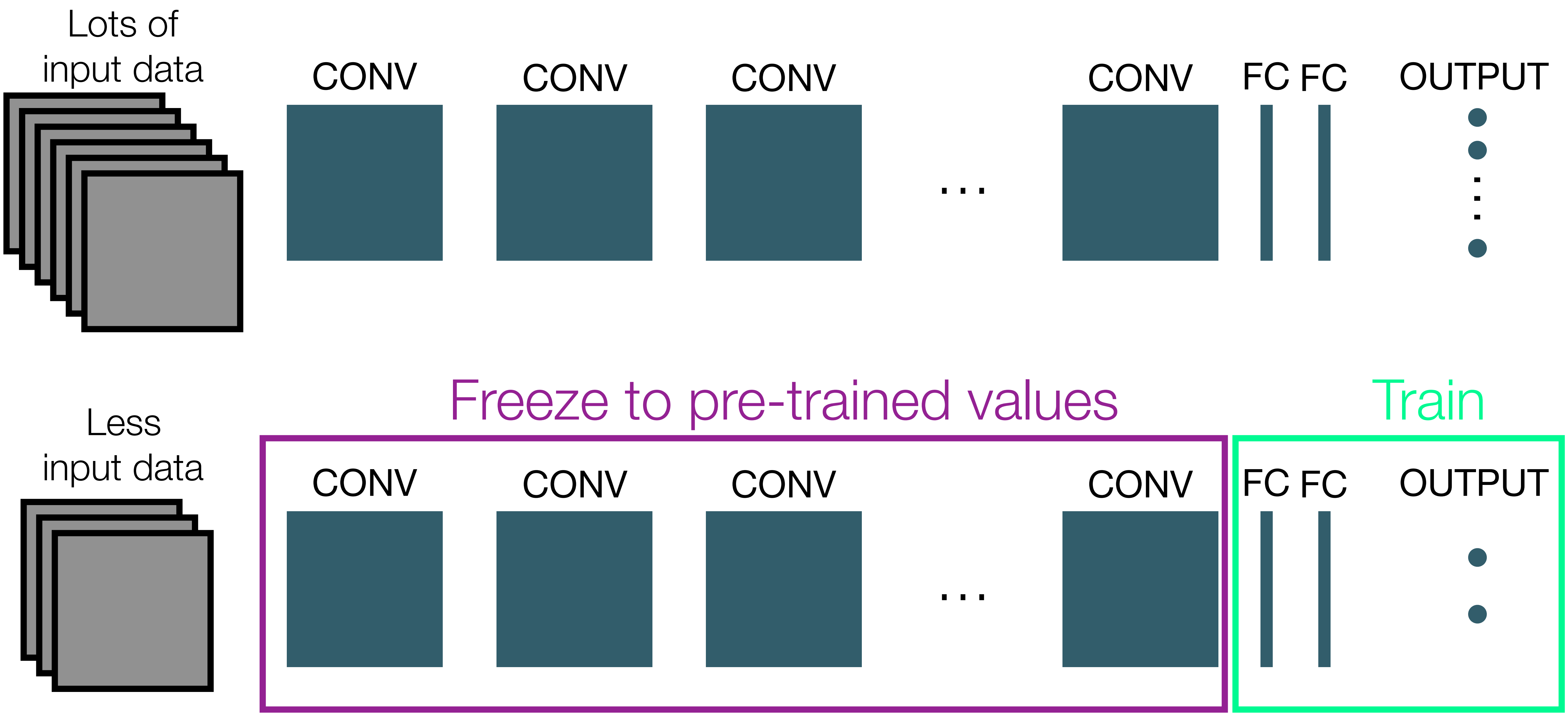
Transfer Learning Techniques for CV

- First train data on large available data set (i.e. imagnet) or better yet, download the pre-trained weights for a specific CNN architecture



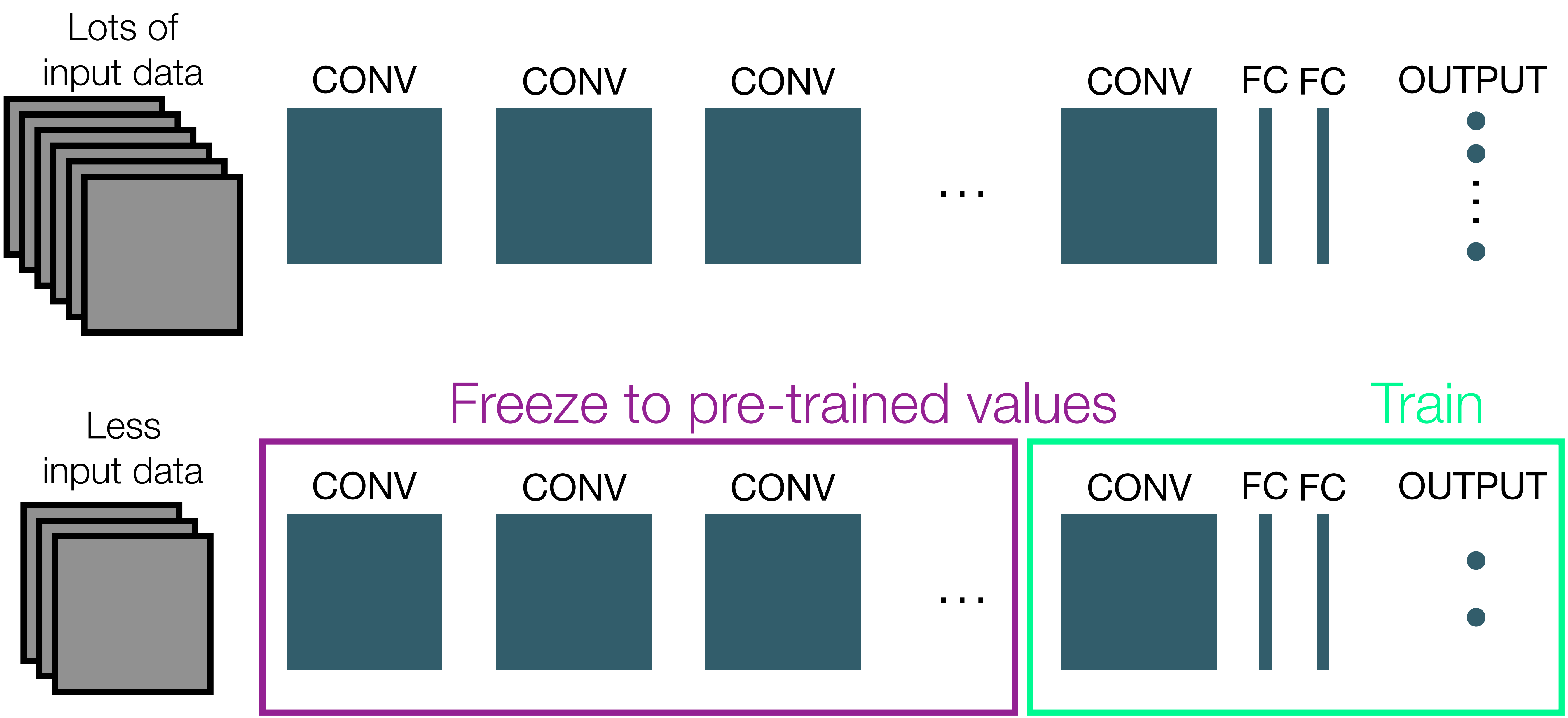
Transfer Learning Techniques for CV

- First train data on large available data set (i.e. imagnet) or better yet, download the pre-trained weights for a specific CNN architecture



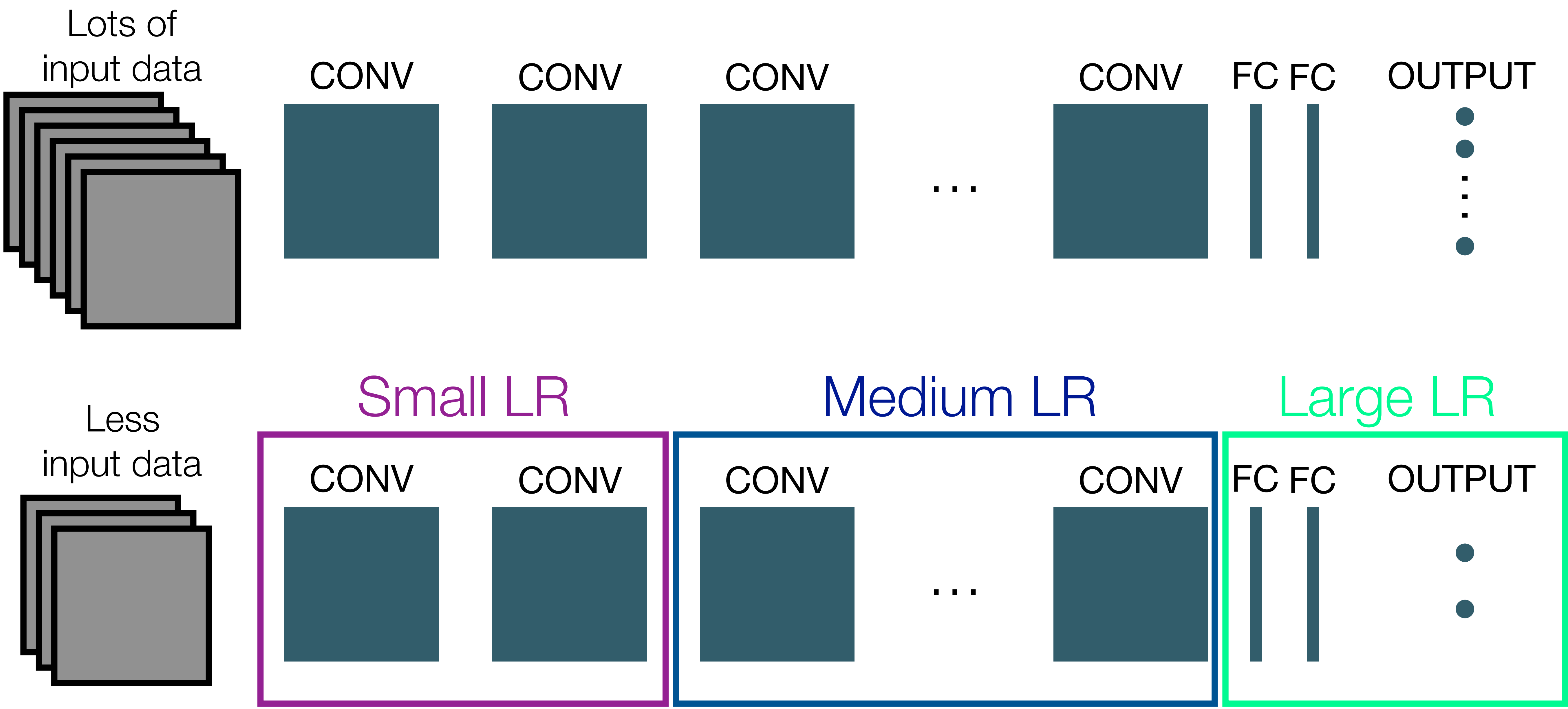
Transfer Learning Techniques for CV

- First train data on large available data set (i.e. imagnet) or better yet, download the pre-trained weights for a specific CNN architecture



Transfer Learning Techniques for CV

- First train data on large available data set (i.e. imagnet) or better yet, download the pre-trained weights for a specific CNN architecture



Simple application on Cats v. Dogs Kaggle Competition

```
# Pre-trained ResNet 50 models with last softmax dense layer removed
base_model = ResNet50(weights='imagenet', include_top=False)

# add Dense layer and sigmoid output layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(1, activation='sigmoid')(x)

# freeze convolution weights and compile model with cross-entropy loss for cat/dog
model = Model(inputs=base_model.input, outputs=predictions)
for layer in base_model.layers: layer.trainable = False
optim = adam(lr=1e-4, decay=1e-6)
model.compile(optimizer=optim, loss='binary_crossentropy', metrics=['accuracy'])

# Run pre-trained model for a few epochs
history = LossHistory()
model.fit_generator(train_generator, train_generator.n // batch_size, epochs=3, workers=4,
                    validation_data=validation_generator,
                    validation_steps=validation_generator.n // batch_size)
```

```
# re-train last convolutional layer
split_at = 157
for layer in model.layers[:split_at]: layer.trainable = False
for layer in model.layers[split_at:]: layer.trainable = True
model.compile(optimizer=optim, loss='binary_crossentropy', metrics=['accuracy'])
```

```
# fit for one epoch
model.fit_generator(train_generator, train_generator.n // batch_size, epochs=1, workers=3,
                    validation_data=validation_generator,
                    validation_steps=validation_generator.n // batch_size)
```

```
Epoch 1/1
359/359 [==============================] - 223s 620ms/step - loss: 0.0617 - acc: 0.9781 - val_loss: 0.0416 - val_acc:
0.9864
```

You can do deep learning too!

- fast.ai courses and forums
- deeplearning.ai courses on Coursera
- Stanford CS courses on youtube

Extra slides

Natural Language Processing (NLP)

- Automatic manipulation and understanding of natural language (i.e. speech, text) by software
- **Text classification** (i.e. sentiment analysis, spam filtering)
- **Language modeling**: learns probabilistic relationship between words and sequences of words (i.e. model *understands* language)
- **Speech recognition**: combination of language modeling and audio data
- **Caption generation**: use language model to generate caption from image
- **Machine translation**: use language model to translate from one language to another
- **Document summary**: use language model to output summary conditioned on entire document

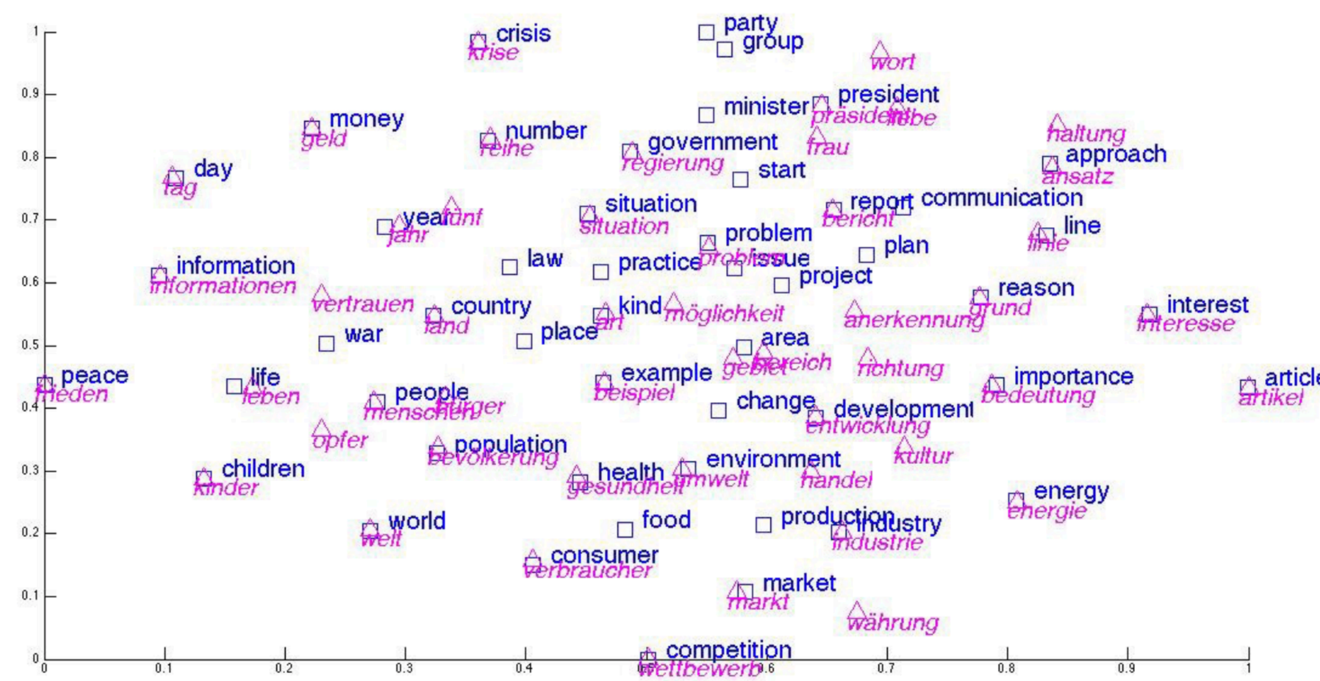
Word embeddings

- Represent words a n-dimensional vector

  cat = [1,6,4,8,9,…,2]
  dog = [5,2,7,4,5,…,8]
  the = [1,2,3,6,3,…,1]

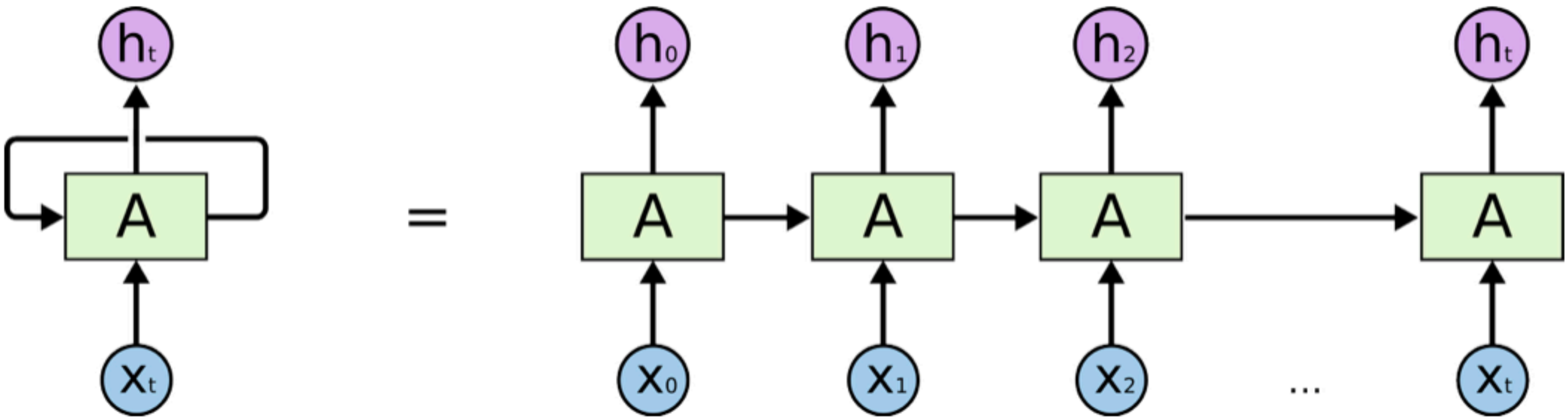
- Can be visualized by projecting into 2-d space where similar words have similar word vectors

- Usually pre-computed for transfer learning tasks



Source: https://medium.com/deeper-learning/glossary-of-deep-learning-word-embedding-f90c3cec34ca

Recurrent Neural Networks (RNN) for NLP



- RNNs are used for sequences of data where there is a temporal order
- Each component in the loop is similar to a standard NN but the activations from that layer (along with the next element in the sequence) are then passed to the next block
- The architecture of the block (GRU, LSTM, etc) can help retain state over long sequences

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Transfer Learning for NLP (1)

- Unlike CV, transfer learning in NLP is relatively new
- New FitLaM method seems promising
  - Leverage large amounts of available data (imagnet)
  - Utilize task, which can be optimized independently (multi-class image classification)
  - Rely on a single model that can be used as-is for most NLP tasks (CNNs)
  - Easy to use in practice (pre-trained networks in Keras/Pytorch)

Howard and Ruder 2018, Fine Tuned Language Models for Text Classification

Transfer Learning for NLP (2)

- Step 1: General domain LM training (uses wikitext-103)
  - Use standard LSTM with highly tuned regularization techniques
  - Weights and code will be made available
- Step 2: Target task LM fine tuning
  - Use gradual unfreezing as in CV
  - Use cosine annealing over epoch and warm-up reverse annealing before unfreezing
- Step 3: Target talk classifier fine-tuning
  - Use discriminative fine tuning with different learning rates at different layers

Howard and Ruder 2018, Fine Tuned Language Models for Text Classification

Transfer Learning for NLP (3)

	Model	Test		Model	Test
IMDb	BCN+Char+CoVe (McCann et al., 2017)	91.8	TREC-6	BCN+Char+CoVe (McCann et al., 2017)	95.8
	oh-LSTM (Johnson and Zhang, 2016)	94.1		TBCNN (Mou et al., 2015)	96.0
	Virtual (Miyato et al., 2016)	94.1		LSTM-CNN (Zhou et al., 2016)	96.1
	FitLaM (Ours)	***95.4***		***FitLaM (ours)***	***96.4***

Table 2: Test accuracy scores on two text classification datasets used by McCann et al. (2017).

	AG-News	DBpedia	Yelp-bi
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64
FitLaM (ours)	**5.01**	**0.80**	**2.16**

Table 3: Test error rates (%) on three text classification datasets used by Johnson and Zhang (2017).

- Still a work in progress that requires a decent amount of hand-tuning and tricks
- If task-specific data has words not in general model vocabulary, what do you do? (Howard suggests setting to mean of other embeddings…)

Howard and Ruder 2018, Fine Tuned Language Models for Text Classification