DataSci 420

lesson 10: deep learning (part 2)

Seth Mottaghinejad

today's agenda

- **convolutional neural networks**
  - applications
  - motivations
  - basic architecture
- **recurrent neural networks**
  - applications
  - motivations
  - basic architecture

image processing applications

- **facial** recognition
- **satellite image** analysis in farming or environmental studies
- optical character recognition
- sound analysis through **audiogram**
- **sentiment analysis** can rely on 1-d convolutions
- **image tagging**
- **anomaly detection** in video images

why use CNNs

- **convolutional neural networks** (CNNs) have dethroned all other image segmentation benchmarks
- image **segmentation** = image **localization** + **classification**
- CNNs preserve **locality** of pixels in an image
- CNNs reduce the connections so we can go deeper: this is done though **shared weights** used by **filters** (aka **kernels**)
- using **transfer learning**, we can directly use features learned from a **pre-trained model**, or **fine-tune** them

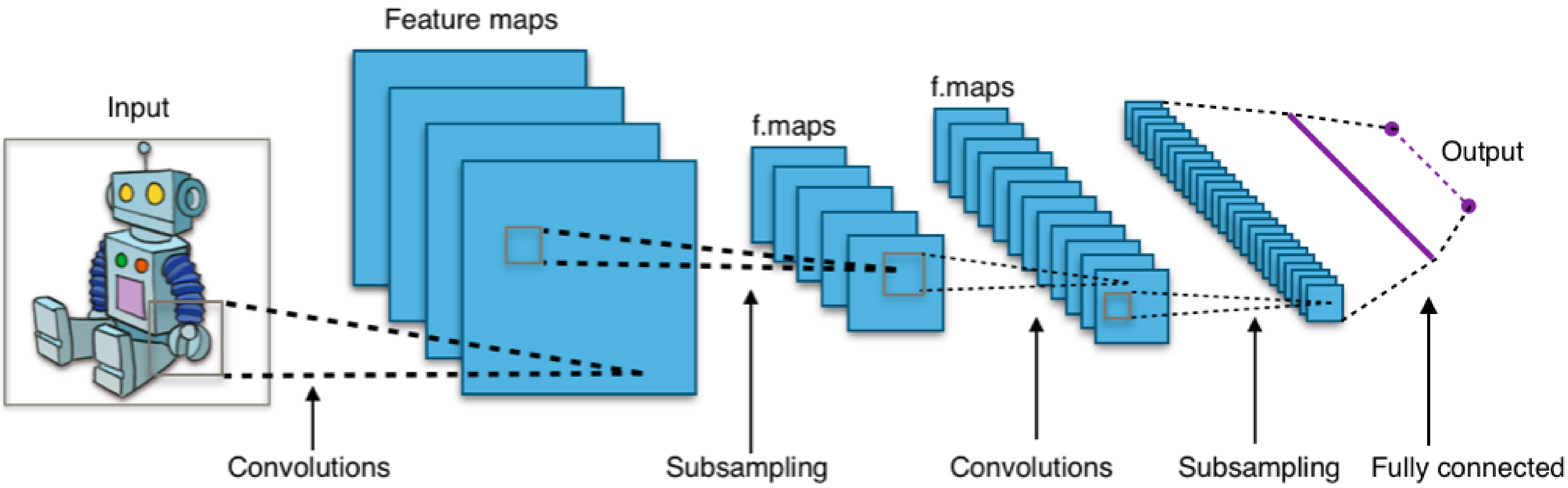


image source: wikipedia.org

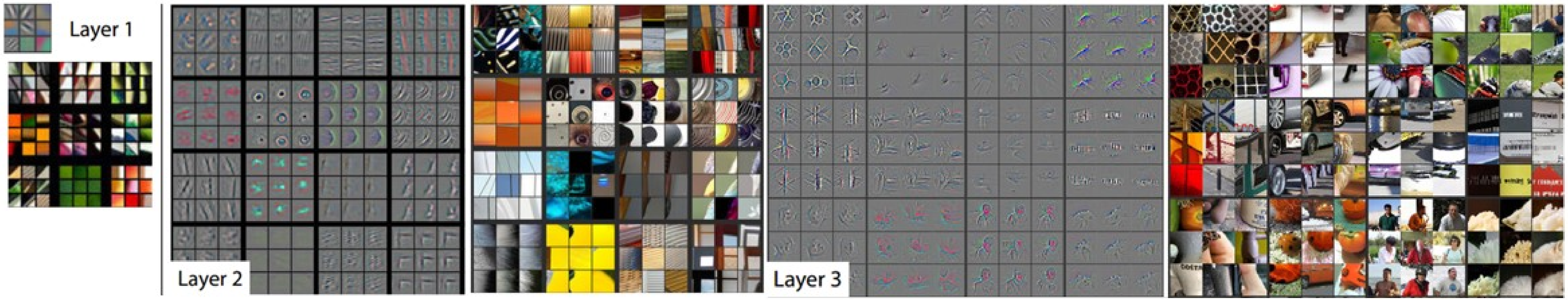


image source: [Visualizing and Understanding Convolutional Neural Networks]

image processing with NNs

- we use **convolutional filters** to process images: blur, sharpen, invert, detect lines and edges, detect colors, etc.
- CNNs **learn** the convolutional filters that should be applied **layer by layer** in order to detect what's in an image
- deeper layers can detect more complex shapes, allowing for a deeper levels of **feature abstraction**
- new models architectures are tested against **benchmark datasets** such as ImageNet (~15 million labeled images from 22K categories)

Image Classification on ImageNet

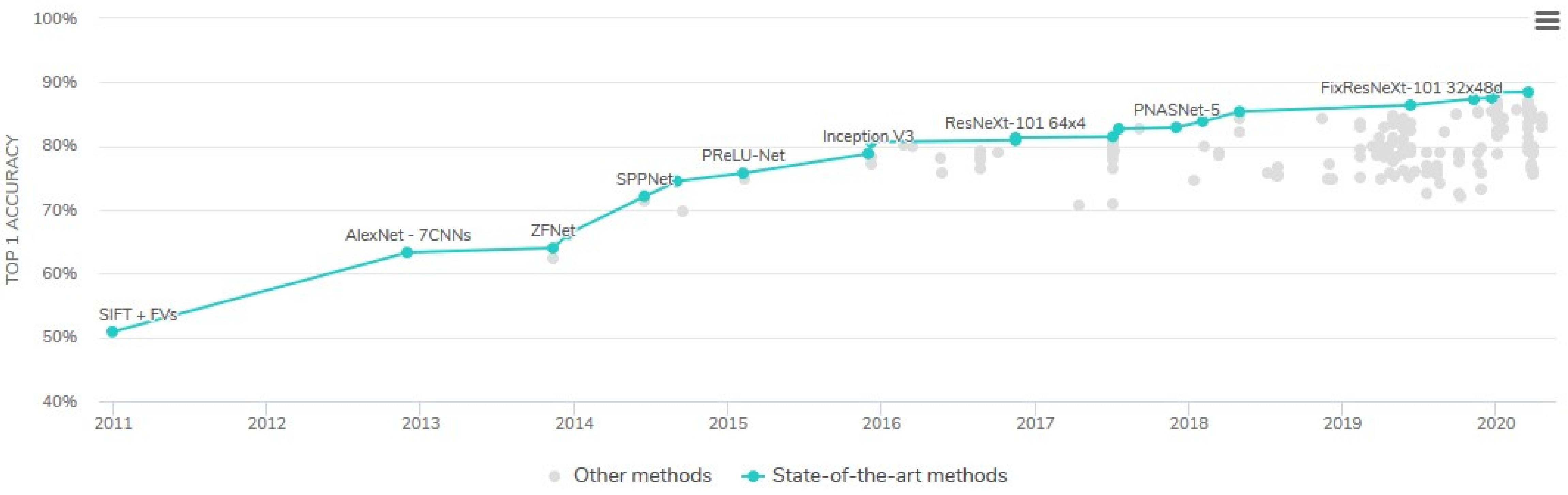


image source: paperswithcode.com

break time

natural language processing (NLP) applications

- machine **translation**
- **speech recognition** for virtual assistants
- question answering / conversation modeling
- **named entity** recognition and extraction
- generating image **caption**
- word or sentence completion (called **language modeling**): in this case the data labels itself: **word + context**

Leaderboard

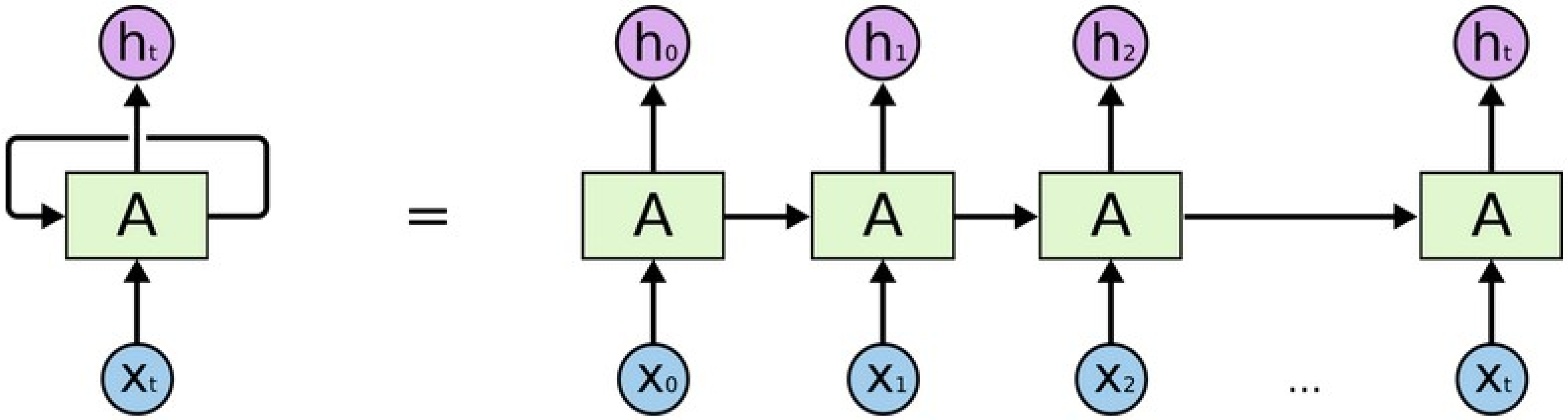
SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.578	92.978
3 Mar 12, 2020	ALBERT + DAAF + Verifier (ensemble) *PINGAN Omni-Sinitic*	90.386	92.777
4 Jan 10, 2020	Retro-Reader on ALBERT (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.115	92.580
5 Nov 06, 2019	ALBERT + DAAF + Verifier (ensemble) *PINGAN Omni-Sinitic*	90.002	92.425
6 Sep 18, 2019	ALBERT (ensemble model) *Google Research & TTIC* https://arxiv.org/abs/1909.11942	89.731	92.215

image source: [The Stanford Question Answering Dataset]

why use RNNs

- recurrent neural networks (RNNs) can work well with **sequential data** where order matters
- we **unroll** it for n steps and end up with regular a deep neural network, whose weights are shared across time steps
- we can teach them to learn long term information, solving the **vanishing gradient problem** (see LSTMs or GRUs)
- we can improve performance by using **dense representations** (like **word2vec**) instead of **sparse ones** (one-hot encoding)



An unrolled recurrent neural network.

image source: Understanding LSTMs

GRUs

- **gated recurrent units** can combine new information x_t with internal state h_{t-1} from the previous time step
- $z_t = \sigma(W_z x_t + U_z h_{t-1})$ is the **update gate**
- $r_t = \sigma(W_r x_i + U_r h_{t-1})$ is the **reset gate**
- **gates** use **sigmoid** activation σ and **element-wise product** $\odot$
- $\tilde{h} = \tanh(W x_t + r_t \odot U h_{t-1})$ is the **current memory**
- $h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$ is the **internal state** (output)

GRU architecture

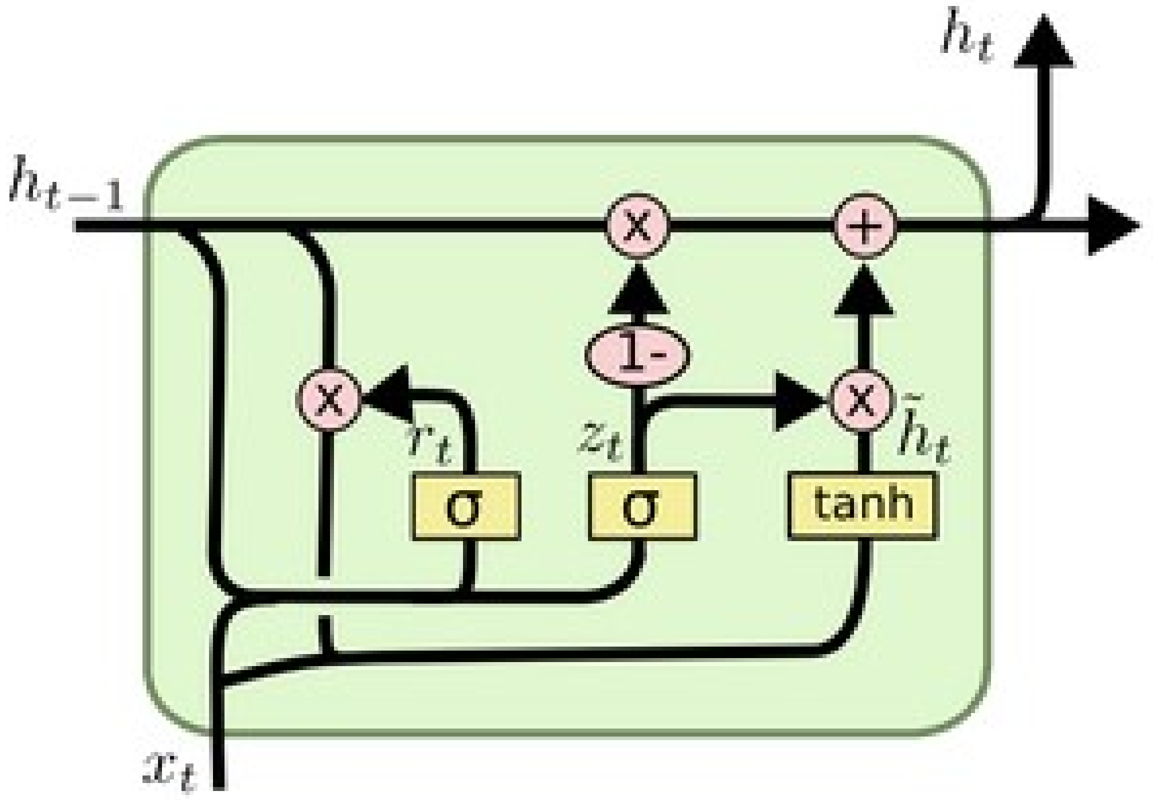


image source: [Understanding LSTMs]

the end