19. Topics in deep learning

- What is deep learning?
- Shining examples of success
- Some more interesting stories

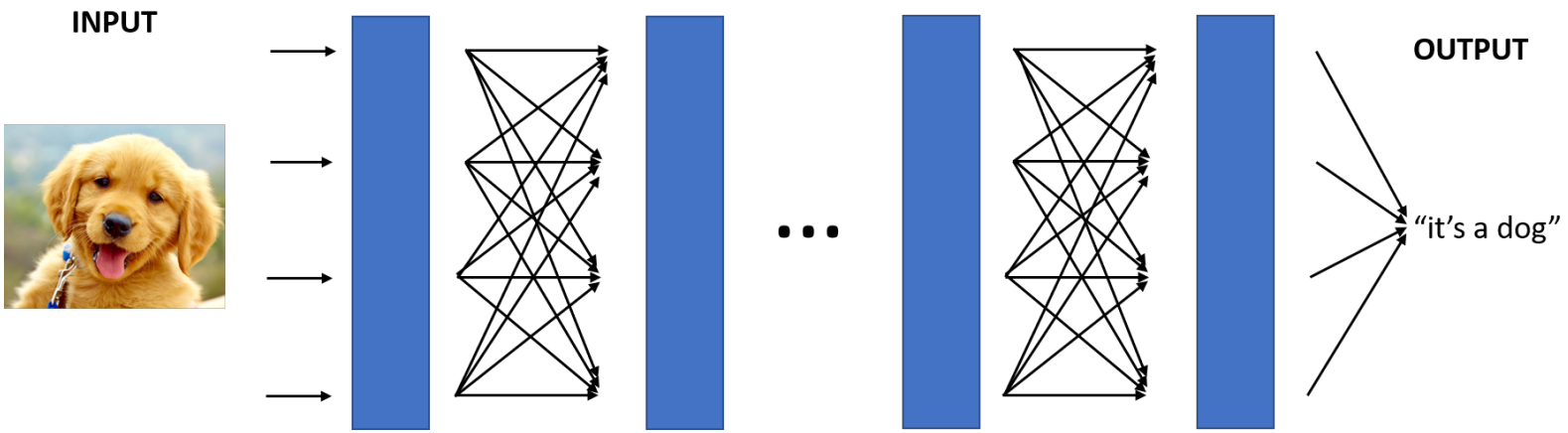
This is a “for fun” lecture

- Get some popcorn
- Put your feet up
- Enjoy!

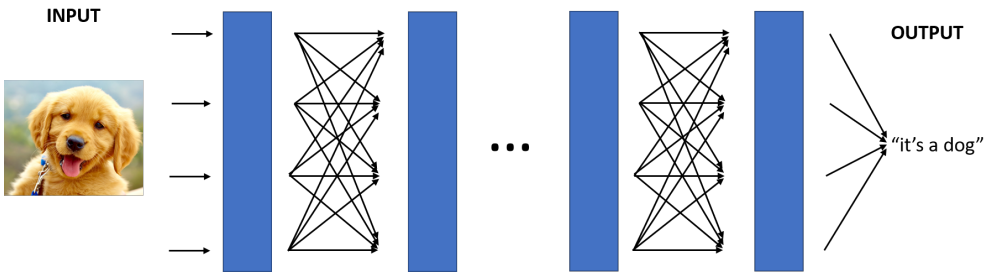
Deep learning models

What is a deep neural network? (DNN)

Simplest case: multilayer perceptron



Simple deep model



- Neurons or states store values based on input
- Connections or weights store model parameters
- At each layer:

$$s_{k+1} = \sigma(W_k s_k), \quad \sigma(\xi) = \frac{1}{1+e^{-\xi}}, \qquad s_0 = x^{(t)}$$

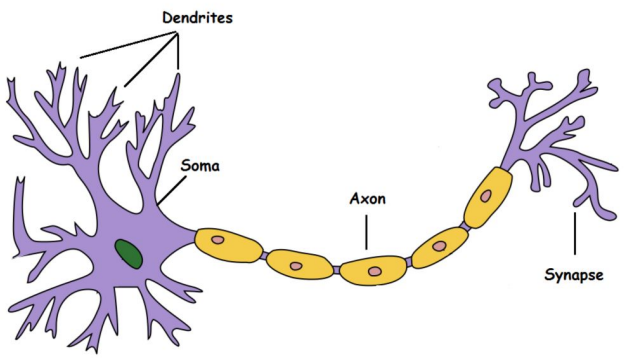
- Output:

$$\hat{y}^{(t)} = \begin{cases} \tanh(W_L s_{L-1}) & \text{binary classification} \\ \dfrac{e^{W_L s_{L-1}}}{\sum_{i=1}^{K} e^{(W_L s_{L-1})[i]}} & \text{multiclass classification} \end{cases}$$

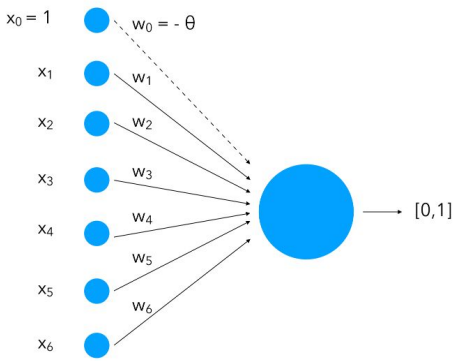
- Layers $s_2, ..., s_L$ are <u>hidden layers</u>.

Perceptron

The brain



Perceptron



$$\hat{y} = f(Wx)$$

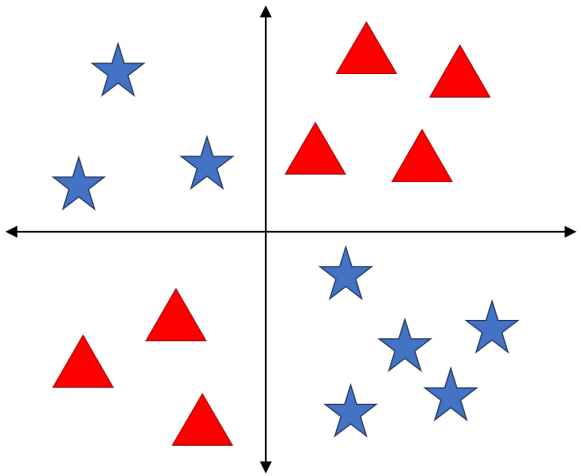
- McCulloch-Pitts (1943): f is OR, AND, GREATER THAN
- Rosenblatt (1957): f is Heaviside function ($f(Wx) = \mathbf{sign}(Wx)$)
- Minsky and Papert (1969): f is sigmoid function

Blog of Maël Fabien: `https://maelfabien.github.io/deeplearning/Perceptron`

Shallow vs deep

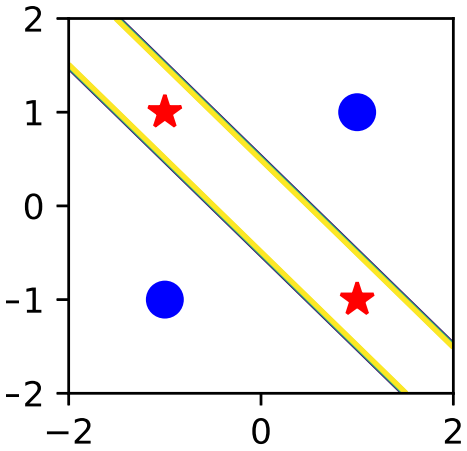
A one layer MLP (no hidden layers) = logistic regression

- nice convex problem
- linear classifier
- one-layer MLP cannot fit: $\rightarrow$
- modeling capabilities severely limited!



2 layers = enough!

$$s = \max\left(\begin{bmatrix} x[1] + x[2] + c \\ x[1] + x[2] - c \end{bmatrix}\right), \quad \hat{y} = [s[1] - s[2]]$$



Universal approximation theorem: A neural network with either

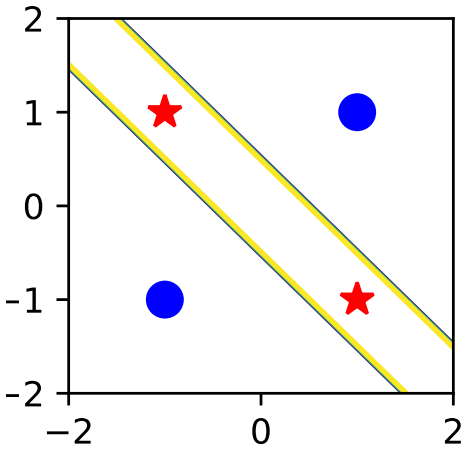
- ≥ 2 layers, where the hidden layer has arbitrarily large width, or
- bounded with but infinite depth

can approximate any continuous function with arbitrary precision.

Cybenko '89, Hornik '91, Leshno '93, Pinkus '99, Lu '17, Hanin and Sellke '18, Kidger and Lyons '20

2 layers = enough!

$$s = \max\left(\begin{bmatrix} x[1] + x[2] + c \\ x[1] + x[2] - c \end{bmatrix}\right), \quad \hat{y} = [s[1] - s[2]]$$



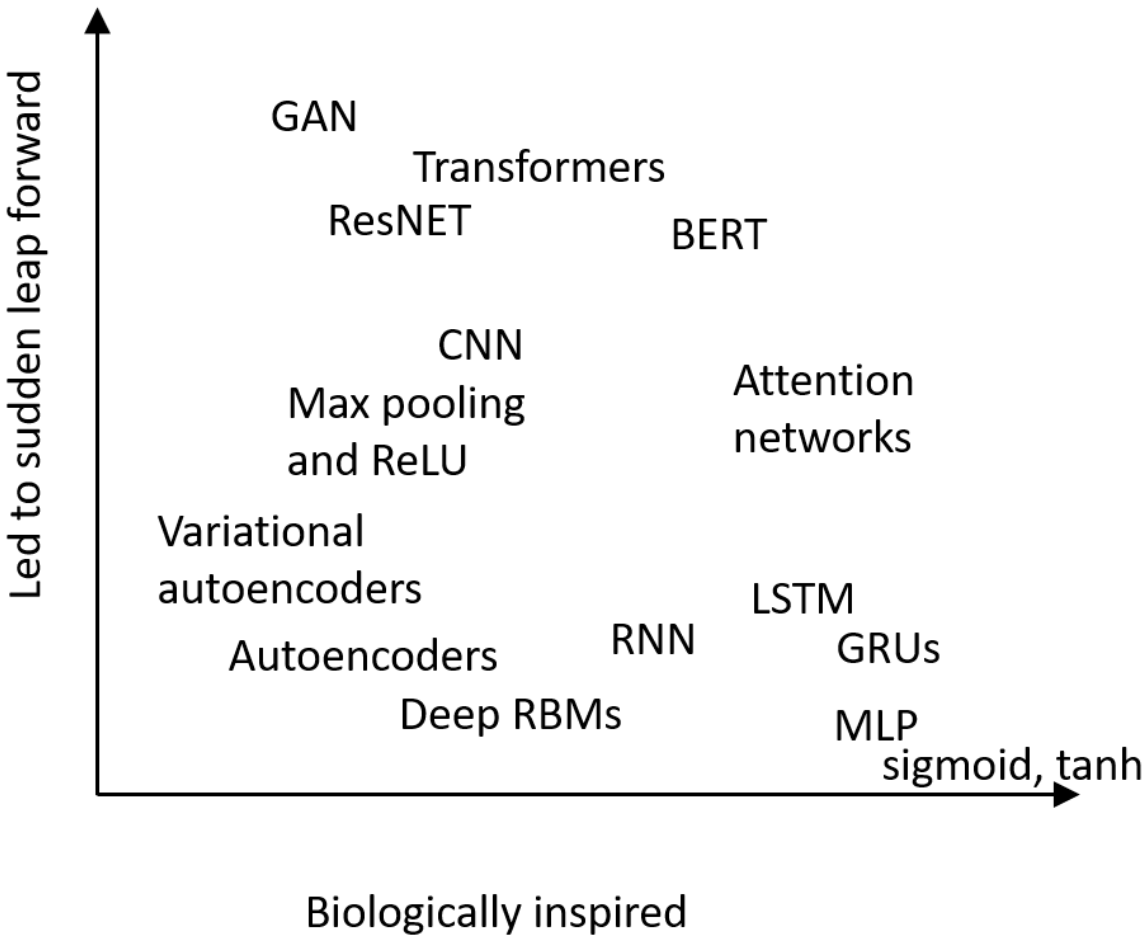
Universal approximation theorem: A neural network with either

- ≥ 2 layers, where the hidden layer has arbitrarily large width, or
- bounded with but infinite depth ("better")

can approximate any continuous function with arbitrary precision.

Cybenko '89, Hornik '91, Leshno '93, Pinkus '99, Lu '17, Hanin and Sellke '18, Kidger and Lyons '20

Tiny idea → huge field

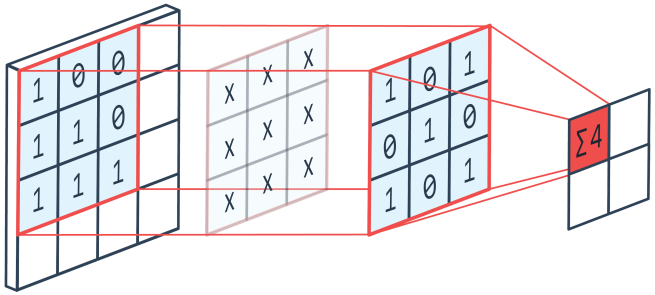


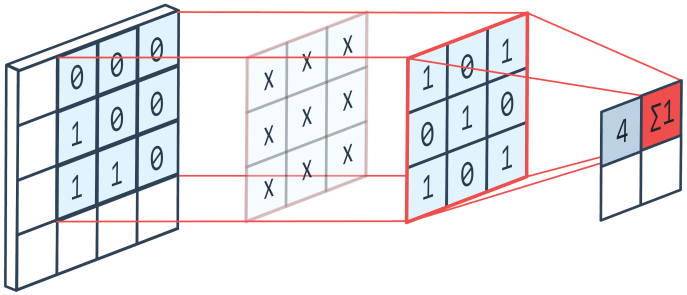
Convolutional neural network (CNN)

Convolution

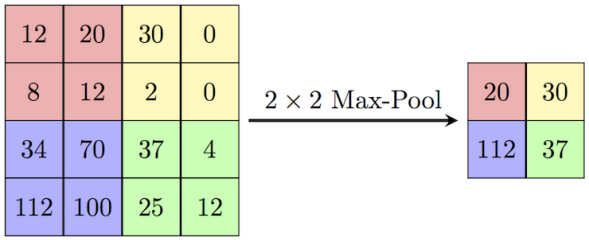
$$f(t) * g(t) := \int_{-\infty}^{\infty} f(\tau) g(t-\tau) dt$$

Convolution on an image with 3x3 filter





Reduce with max operation (max pooling)



Images: peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/2d-convolution-block,computersciencewiki.org/index.php/Max-pooling_

Convolutional neural network (CNN) example

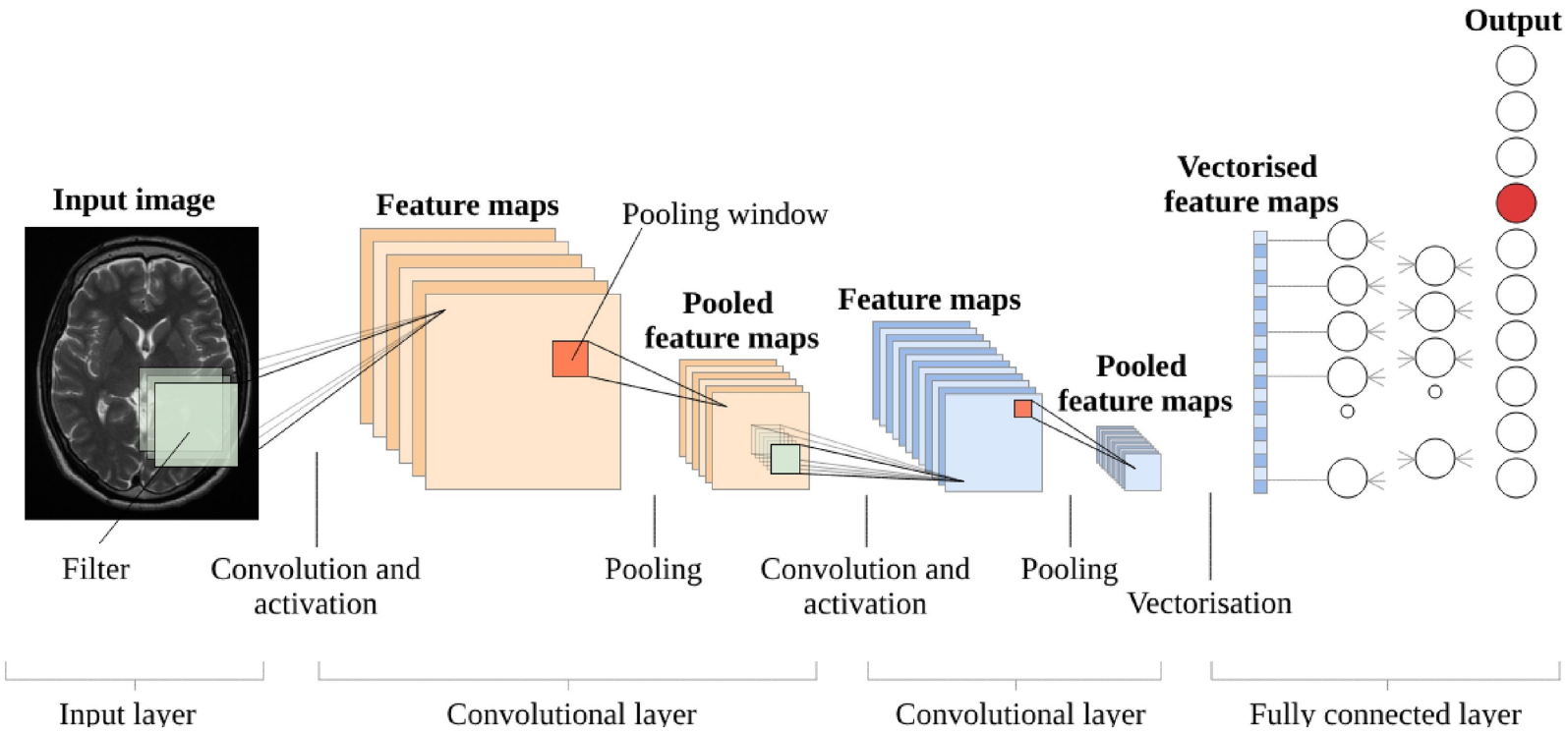
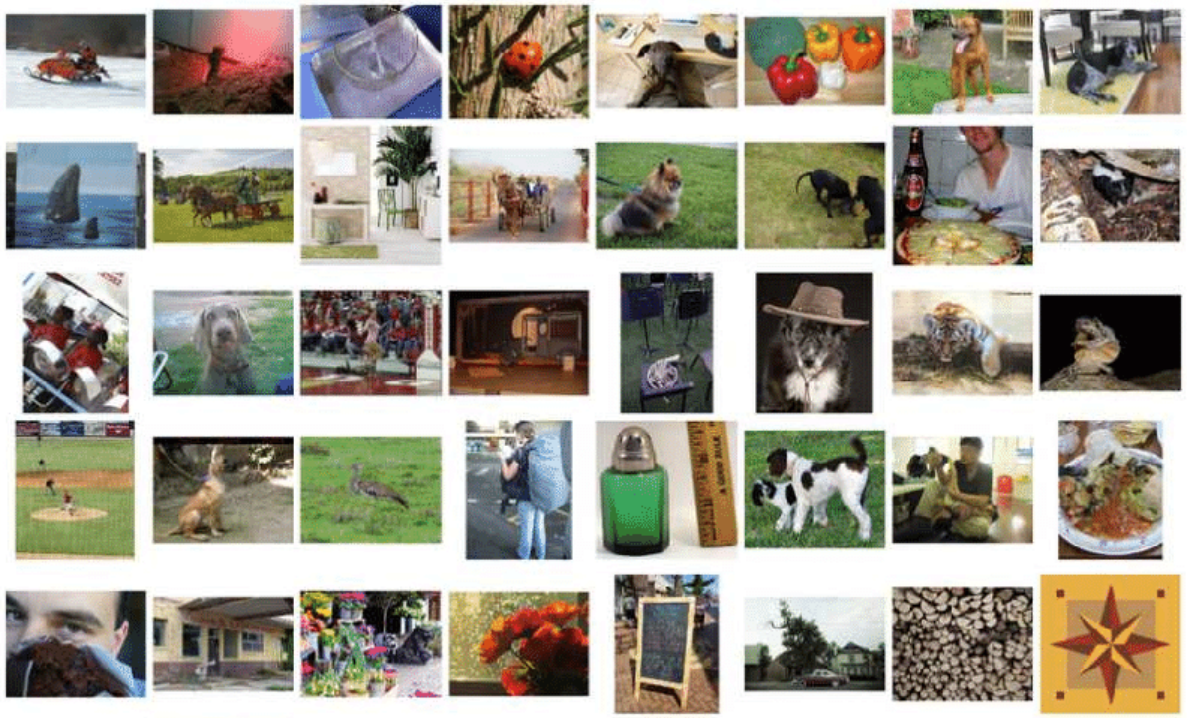


Image courtesy of Lundervold, Lundervold '19

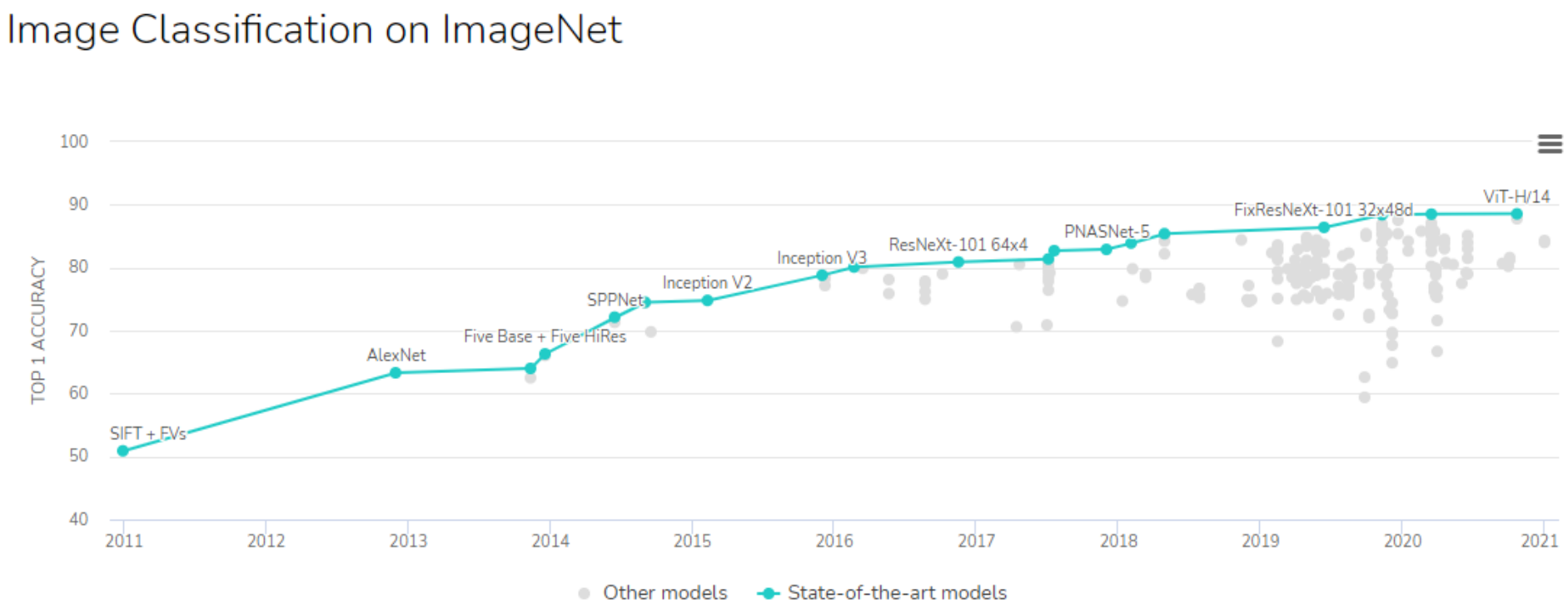
ImageNet

- $>$ 14 million images, hand annotated
- bounding boxes provided for $>$ 1 million images
- $>$ 20,000 categories, several hundred images per category



'09 FeiFei Li and Christiane Fellbaum in CVPR

ImageNet



https://paperswithcode.com/sota/image-classification-on-imagenet

Deep learning as optimization problem

$$\min_{W_1,\cdots,W_L} \quad \sum_{i=1}^{m} \underbrace{f_L(W_L f_{L-1}(\cdots f_2(W_2 f_1(W_1 x_i))\cdots), y_i)}_{F_i(W_1,\ldots,W_L)}$$

- f_L may be logistic loss, or multiclass loss
- $(x_1, y_1), ..., (x_m, y_m)$ training data/labels
- nonconvex, L-smooth (but L is hard to compute)
- stochastic / incremental gradient descent

$$W_k^{(t+1)} = W_k^{(t+1)} - \alpha \nabla_{W_k} F_{i_t}(W_1^{(t)}, ..., W_L^{(t)}), \quad \forall k, \quad i_t \sim \text{Uniform}\{1, ..., m\}$$

Calculating incremental gradients

$$\min_{W_1,\cdots,W_L} \quad \sum_{i=1}^{m} \underbrace{f_L(W_L f_{L-1}(\cdots f_2(W_2 f_1(W_1 x_i))\cdots), y_i)}_{F_i(W_1,\ldots,W_L)}$$

Key idea: gradient descent on nested functions benefits from caching intermediate variables

- Define $s_k^{(t)} = f_k(W_k^{(t)} s_{k-1}^{(t)})$, $s_0^{(t)} = x_{i_t}$
- Jacobian

$$J_{s_k,s_{k-1}} = W_k^T f_k'(W_k s_{k-1}),$$

- Gradient

$$\underbrace{\nabla_{W_k} F_i}_{\text{matrix}} = \underbrace{(\nabla_{s_k} F_i) s_{k-1}^T}_{\text{outer product}}, \qquad \underbrace{\nabla_{s_k} F_i}_{\text{vector}} = \underbrace{J_{s_{k+1},s_k}^T}_{\text{matrix}} \underbrace{\nabla_{s_{k+1}} F_i}_{\text{vector}}$$

Vanishing and exploding gradient problems

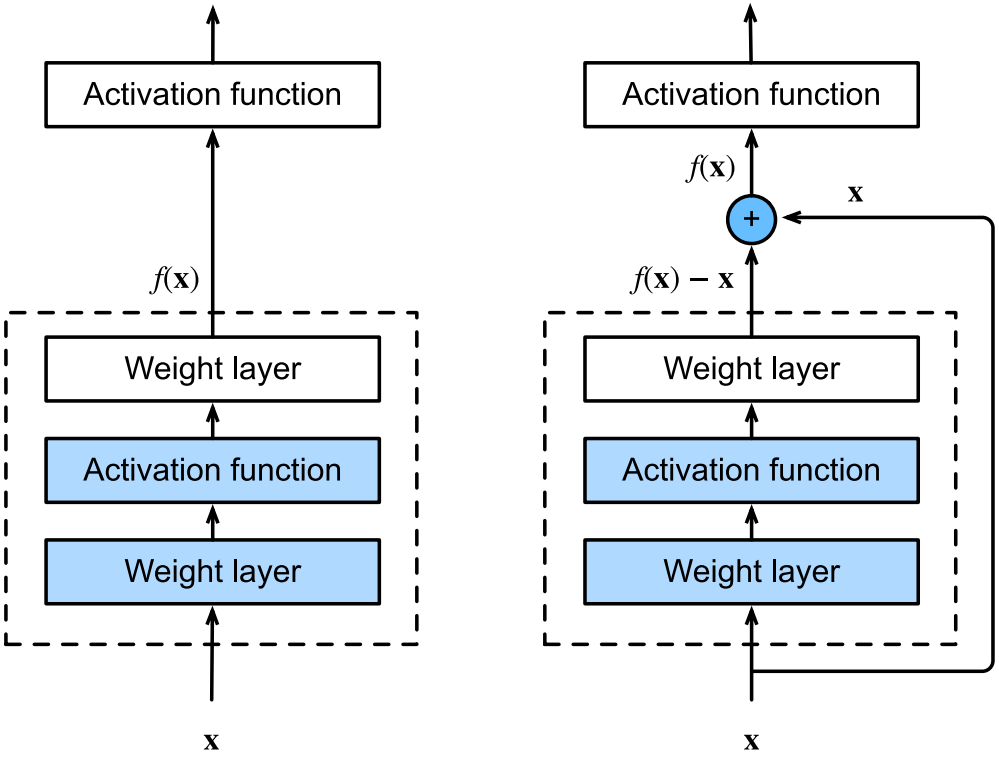
$$\nabla_{s_k} F_i = \underbrace{J^T_{s_{k+1},s_k} J^T_{s_{k+2},s_{k+1}} \cdots J^T_{s_{L-1},s_L}}_{\text{chain of matrix multiplications}} \nabla_{s_L} F_i$$

- exploding gradient: multiply a number $|c| > 1$ a million times $\rightarrow$ huge
  - Solution: gradient clipping

$$J_{s_{k+1},s_k} = \min\{\bar{J}_{s_{k+1},s_k}, 5\}$$

- vanishing gradient: multiply a number $|c| < 1$ a million times $\rightarrow$ 0
  - Solution: ?!?!?! Open area of research
  - ResNet

ResNet



- Extra term helps keep things from vanishing to 0
- Now we can build <u>huge</u> architectures!

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun 2015

image: `https://d2l.ai/chapter_convolutional-modern/resnet.html`

A few more notable gamechangers

Recurrent neural network

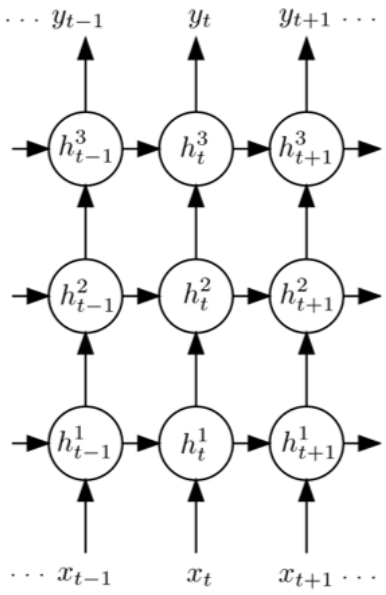


Fig. 3. Deep Recurrent Neural Network

- **Applications**: Anything sequential (stroke detection, speech-to-text, NLP)
- Somehow, never quite achieved the amazing performance as CNNs

Images from: Alex Graves, Navdeep Jaitly and Abdel-rahman Mohamed 2013

LSTM: trains better, but overfits

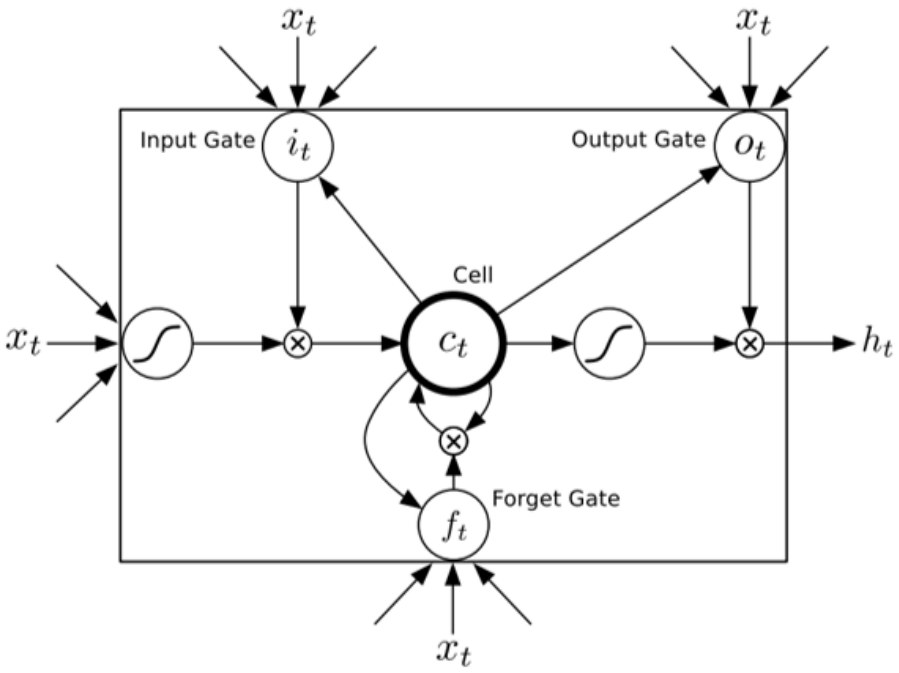


Fig. 1. Long Short-term Memory Cell

Images from: Alex Graves, Navdeep Jaitly and Abdel-rahman Mohamed 2013
Credit goes to: Sepp Hochreiter and Jürgen Schmidhuber, 1995

Bidirectional LSTMs

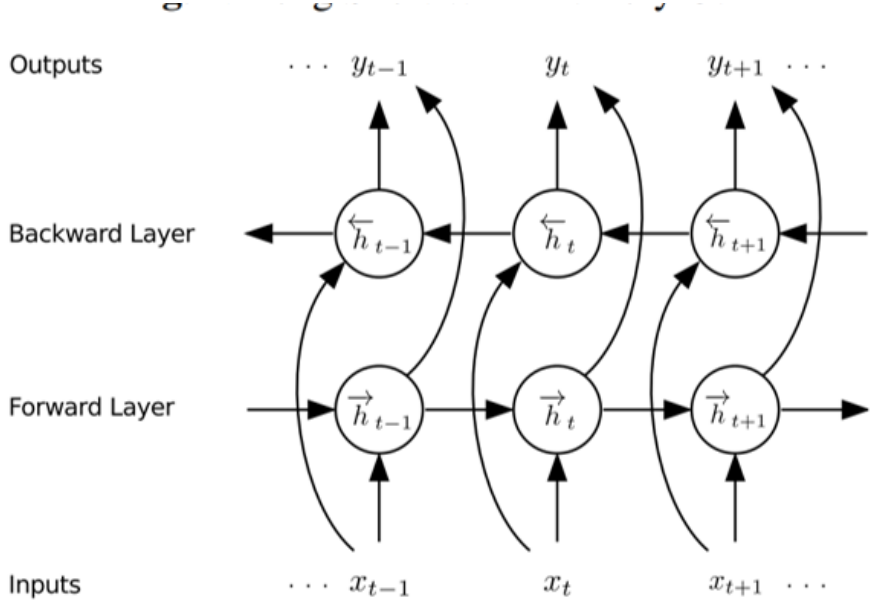


Fig. 2. Bidirectional Recurrent Neural Network

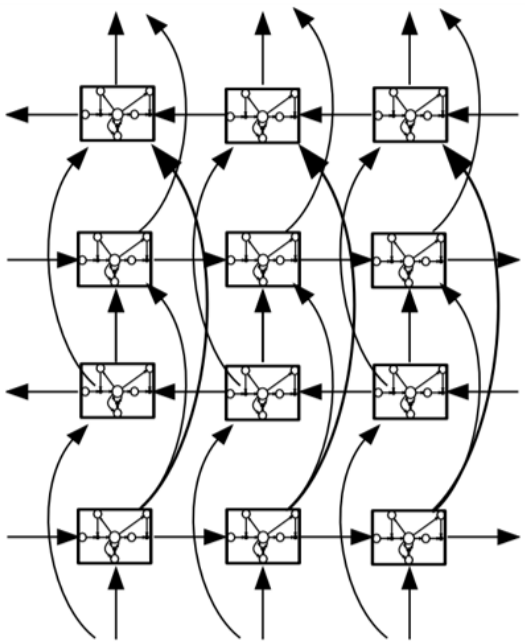


Fig. 4. Deep Bidirectional Long Short-Term Memory Network (DBLSTM)

Alex Graves, Navdeep Jaitly and Abdel-rahman Mohamed 2013

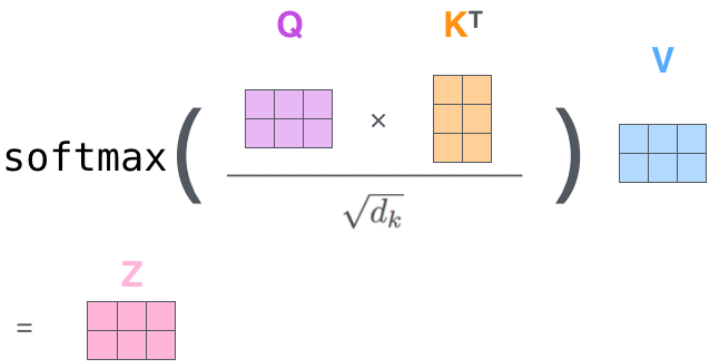
Bidirectional LSTMs outperforming HMMs

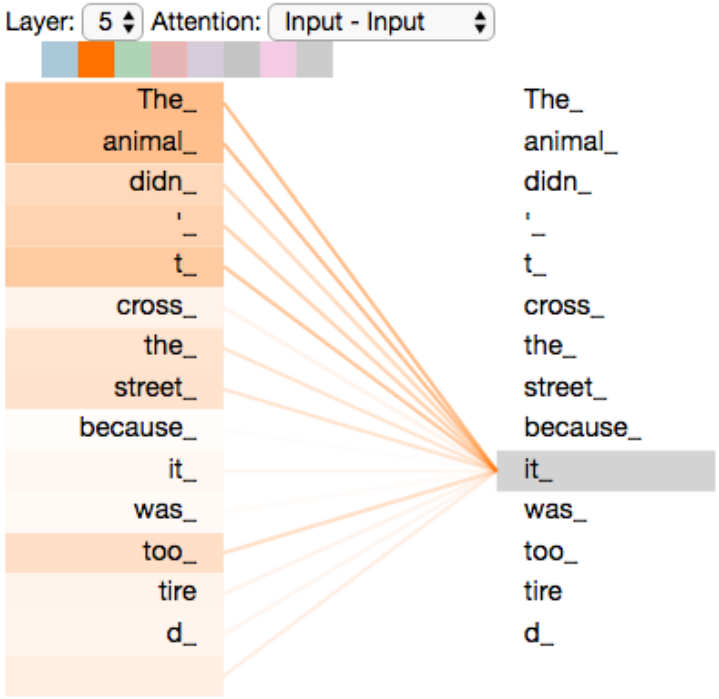
Network	Training Set	Test Set	Epochs
BLSTM	77.4%	69.8%	21
BRNN	76.0%	69.0%	170
BLSTM Weighted Error	75.7%	68.9%	15
LSTM (4 frame delay)	77.5%	65.5%	33
RNN (4 frame delay)	70.8%	65.1%	144
LSTM (0 frame delay)	70.9%	64.6%	15
RNN (0 frame delay)	69.9%	64.5%	120

Word on the street: replaced HMMs as state-of-the-art for speech-to-text

Alex Graves, Santiago Fernández, Jürgen Schmidhuber, '05

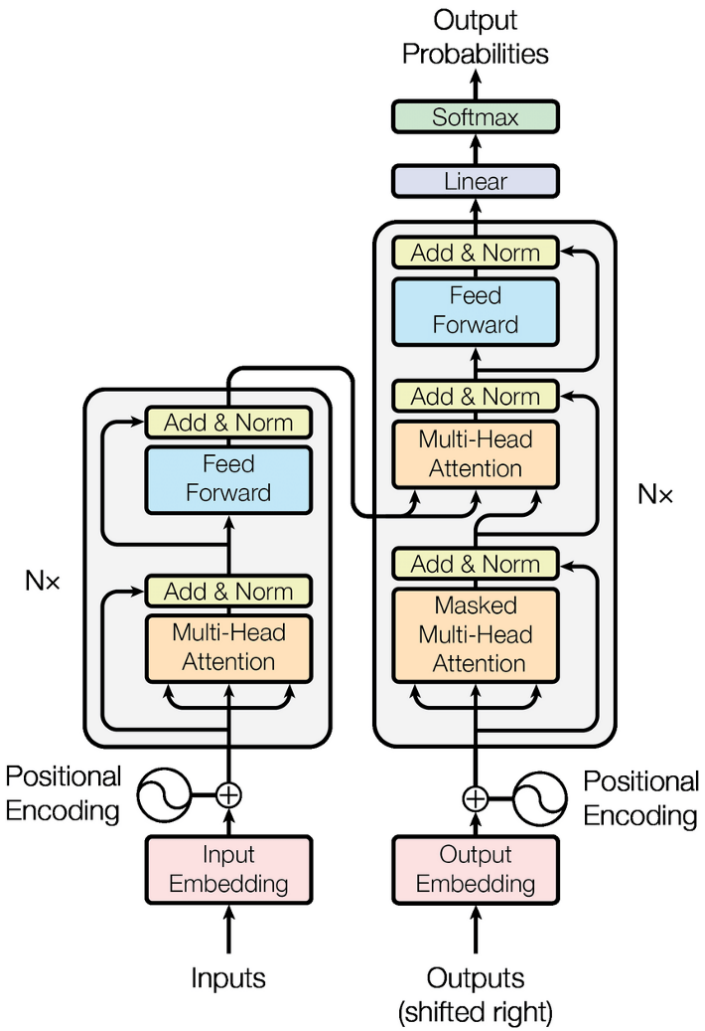
Attention networks





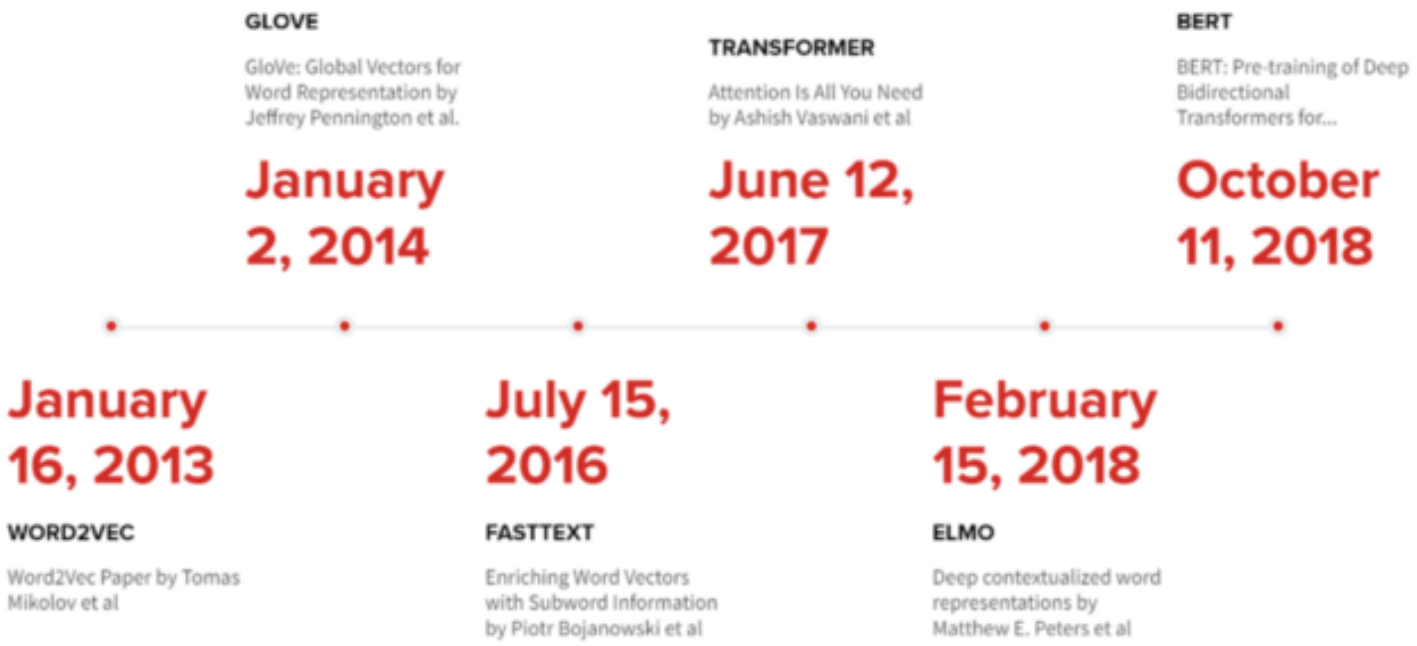
Jay Alammar http://jalammar.github.io/illustrated-transformer/

Transformers (BERT)



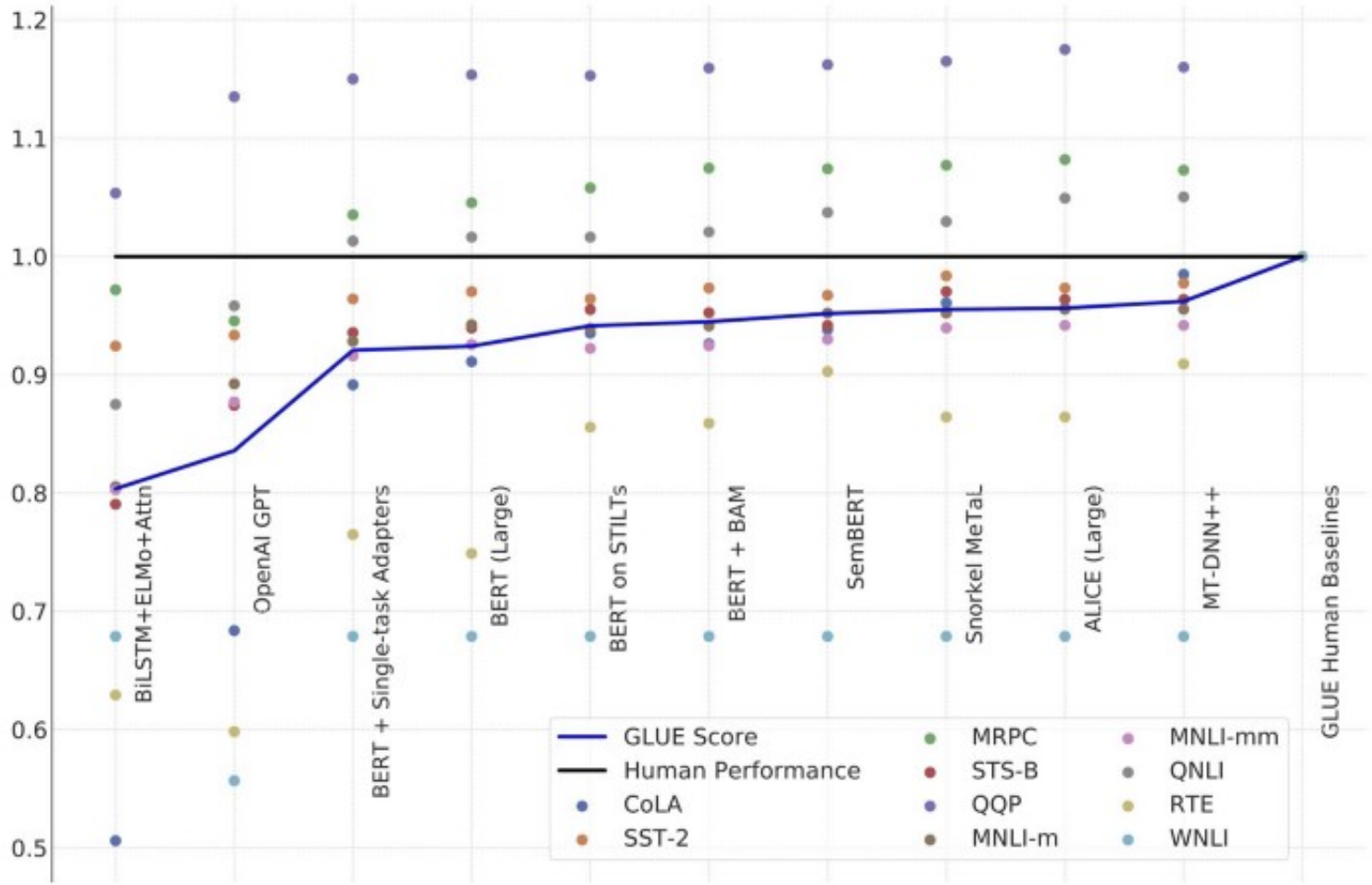
Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova 2019
https://blog.usejournal.com/part1-bert-for-advance-nlp-with-transformers-in-pytorch-

Some history



https://blog.usejournal.com/part1-bert-for-advance-nlp-with-transformers-in-pytorch-357579d63512

Some history

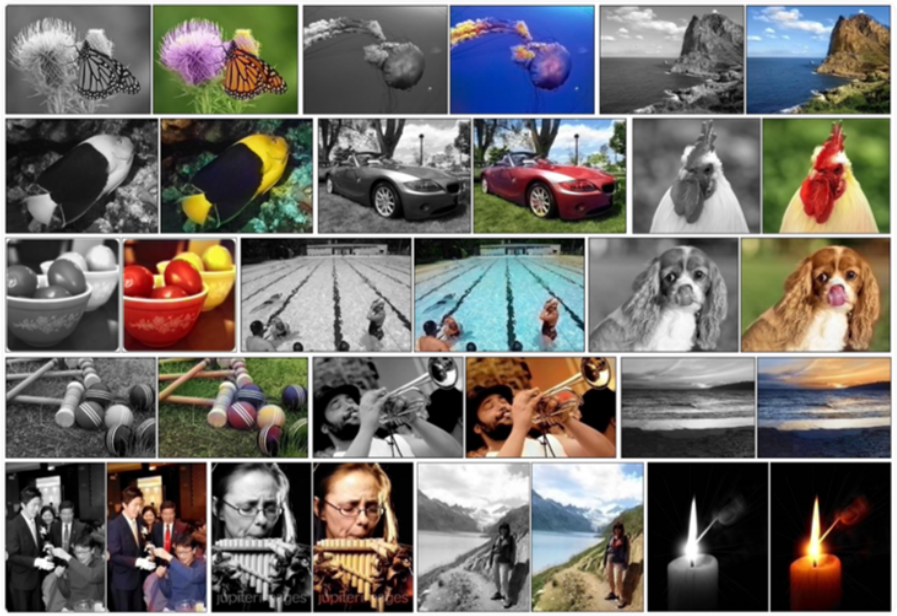


https://medium.com/@wang.alex.c/introducing-superglue-a-new-hope-against-muppetkind-2779fd9dcdd5

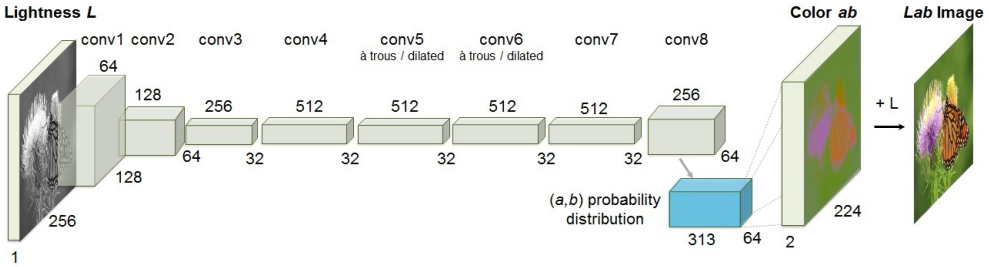
Some stories

Colorization

Zhang, Isola, Efros. Colorful Image Colorization. In ECCV, 2016



Example input grayscale photos and output colorizations from our algorithm. These examples are cases where our model works



Overfitting?

PlaNet - Photo Geolocation with Convolutional Neural Networks

Tobias Weyand[1], Ilya Kostrikov[2], James Philbin[3]

Abstract. Is it possible to determine the location of a photo from just its pixels? While the general problem seems exceptionally difficult, photos often contain cues such as landmarks, weather patterns, vegetation, road markings, or architectural details, which in combination allow to infer where the photo was taken. Previously, this problem has been approached using image retrieval methods. In contrast, we pose the problem as one of classification by subdividing the surface of the earth into thousands of multi-scale geographic cells, and train a deep network using millions of geotagged images. We show that the resulting model, called *PlaNet*, outperforms previous approaches and even attains superhuman accuracy in some cases. Moreover, we extend our model to photo albums by combining it with a long short-term memory (LSTM) architecture. By learning to exploit temporal coherence to geolocate uncertain photos, this model achieves a 50% performance improvement over the single-image model.

Overfitting?

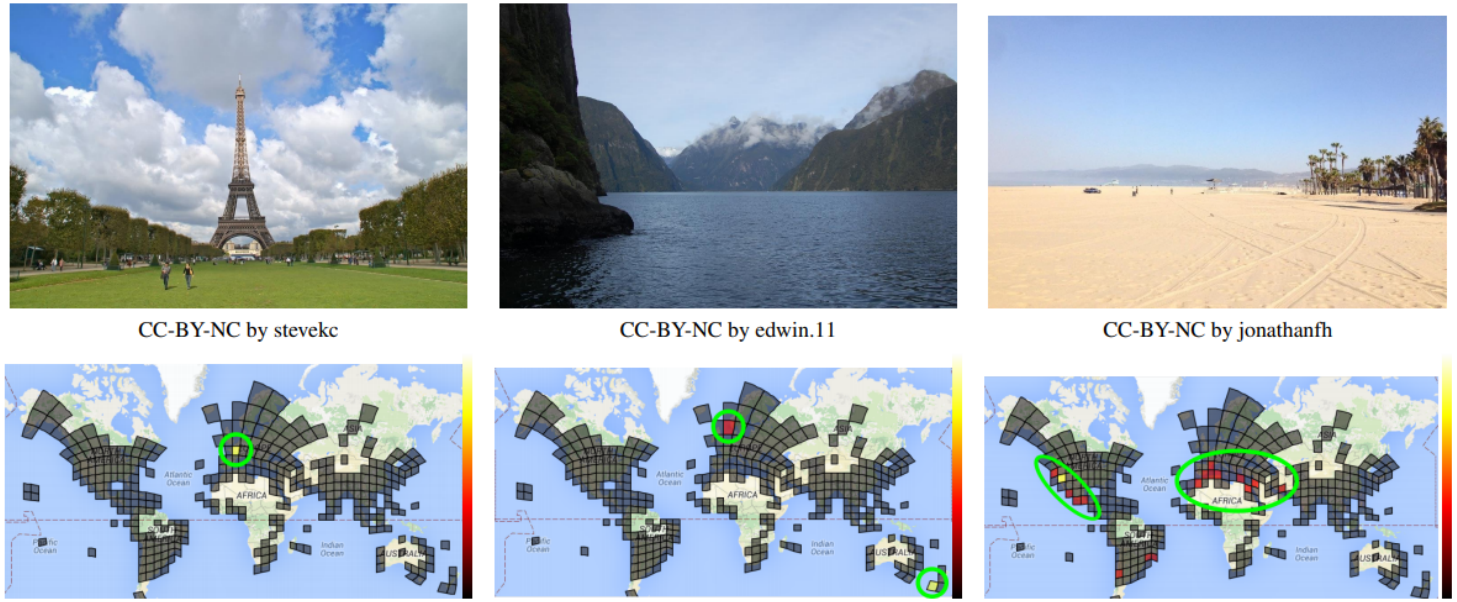


Fig. 1. Given a query photo (top), PlaNet outputs a probability distribution over the surface of the earth (bottom). Viewing the task as a classification problem allows PlaNet to express its uncertainty about a photo. While the Eiffel Tower (a) is confidently assigned to Paris, the model believes that the fjord photo (b) could be taken in either New Zealand or Norway. For the beach photo (c), PlaNet assigns the highest probability to southern California (correct), but some probability is also assigned to places with similar beaches, like Mexico and the Mediterranean. (For visualization purposes we use a model with a much lower spatial resolution than our full model)

Overfitting?

Fig. 6. Examples of images PlaNet localizes correctly. Our model is capable of localizing photos of famous landmarks (top row), but often yields surprisingly accurate results for images with more subtle geographical cues. The model learns to recognize locally typical landscapes, objects, architectural styles and even plants and animals

Overfitting?

Fig. 8. The top-5 most confident images from the Flickr dataset for the S2 cells on the left, showing the diverse visual representation of places that PlaNet learns

Overfitting?

Method	Street 1 km	City 25 km	Region 200 km	Country 750 km	Continent 2500 km
Im2GPS (orig) [19]		12.0%	15.0%	23.0%	47.0%
Im2GPS (new) [20]	2.5%	21.9%	32.1%	35.4%	51.9%
PlaNet (900k)	0.4%	3.8%	7.6%	21.6%	43.5%
PlaNet (6.2M)	6.3%	18.1%	30.0%	45.6%	65.8%
PlaNet (91M)	**8.4%**	**24.5%**	**37.6%**	**53.6%**	**71.3%**

(a)

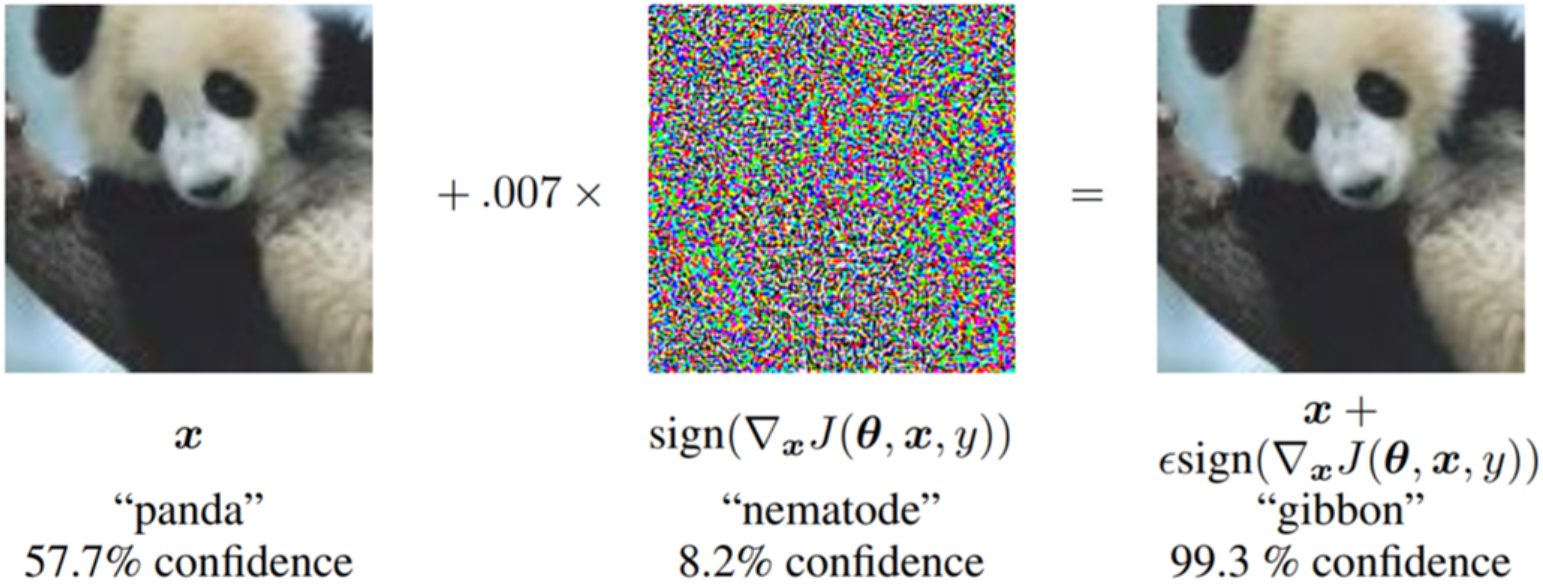
t_1	#classes	med. imgs per class	#model params
100k	214	21,039	23.9M
10k	2,056	2,140	29.1M
5k	3,852	1,225	34.2M
1k	16,307	320	69.3M

(b)

Table 1. (a) Comparison of PlaNet with Im2GPS. Percentages are the fraction of images from the Im2GPS test set that were localized within the given radius. (Numbers for the original Im2GPS are approximate as they were extracted from a plot in the paper.) (b) Parameters of PlaNet models with different spatial resolutions

Robustness and trick-ability

Explaining and harnessing adversarial examples
Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy (2015)



`https://youtu.be/zQ_uMenoBCk`