

Machine Learning

Sep. 23, 2020

Yuta Nakashima

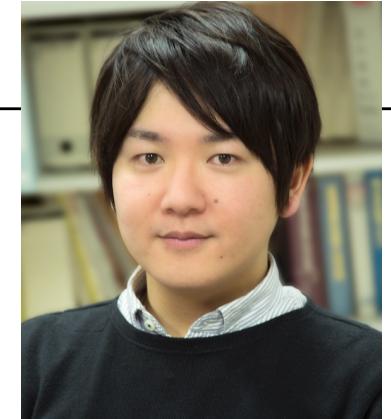


大阪大学データビリティフロンティア機構
Osaka University Institute for Datability Science

About me

- **Yuta Nakashima**

Institute for Datability Science, Osaka University
Associate Professor



- Bio

- -2012: Ph.D Course, Osaka University
- 2012: Visiting Scholar, UNC Charlotte
- 2012-2016: Assistant Prof., Nara Institute for Science and Technology
- 2015-2016: Visiting Scholar, CMU
- 2017- : Current Position

- Research interests

- Computer Vision; CV
- Pattern Recognition; PR
- (Natural Language Processing; NLP)



大阪大学データビリティフロンティア機構
Osaka University Institute for Datability Science

Agenda

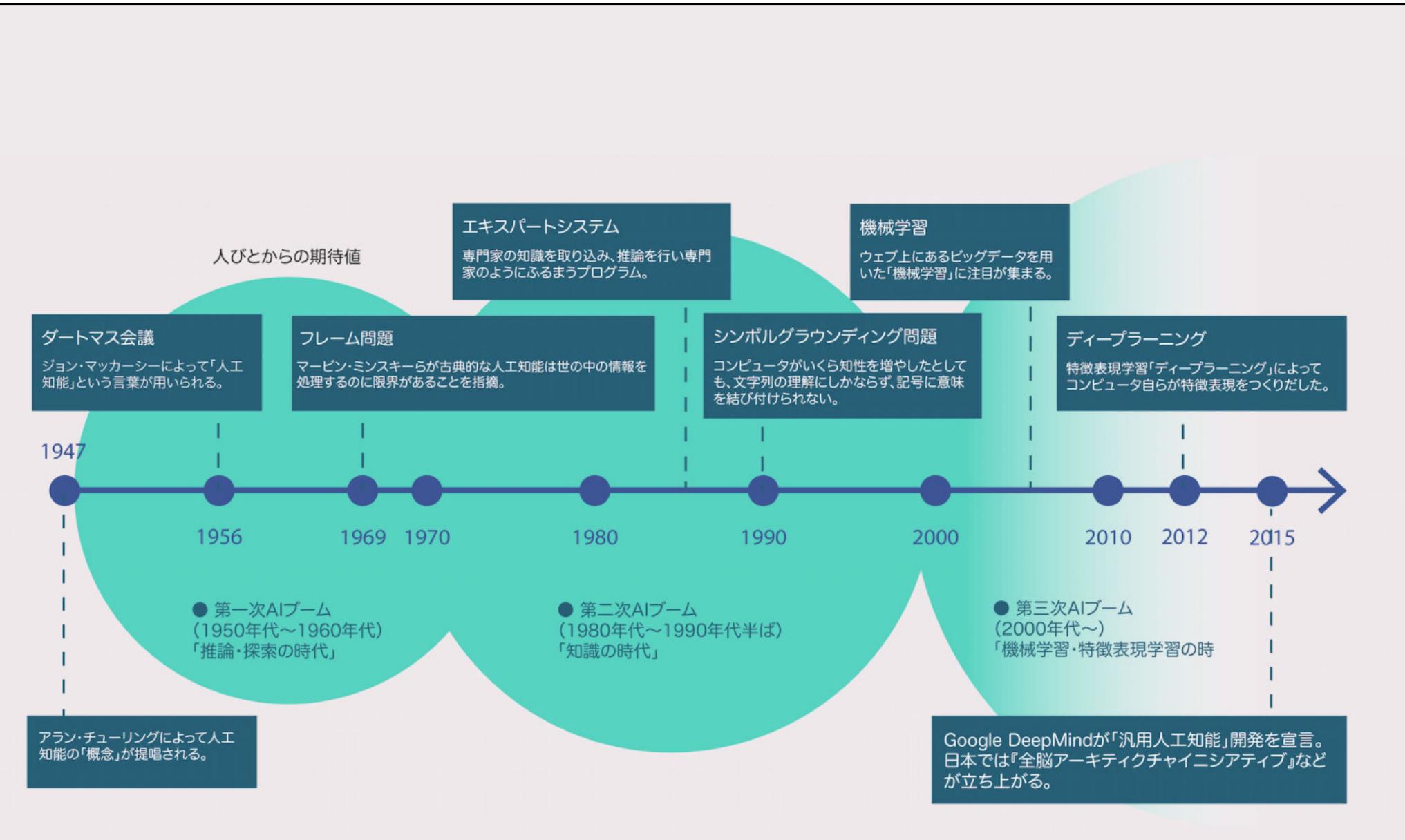
- AI? Machine learning? Deep learning?
- Why deep learning?
- Training a neural network
- Problems in neural networks
- An application to Physics
- Shortcut learning?



**AI?
Machine learning?
Deep learning?**

History of AI

Image taken from: <http://ja.catalyst.red/articles/ai-infographic-01/>



Relationships among AI-related fields

Artificial Intelligence, AI

- Rule bases
- Expert systems

Machine Learning, ML

- Support vector machine
- Logistic regression
- Ridge regression
- (shallow) neural networks

Representation Learning, RL

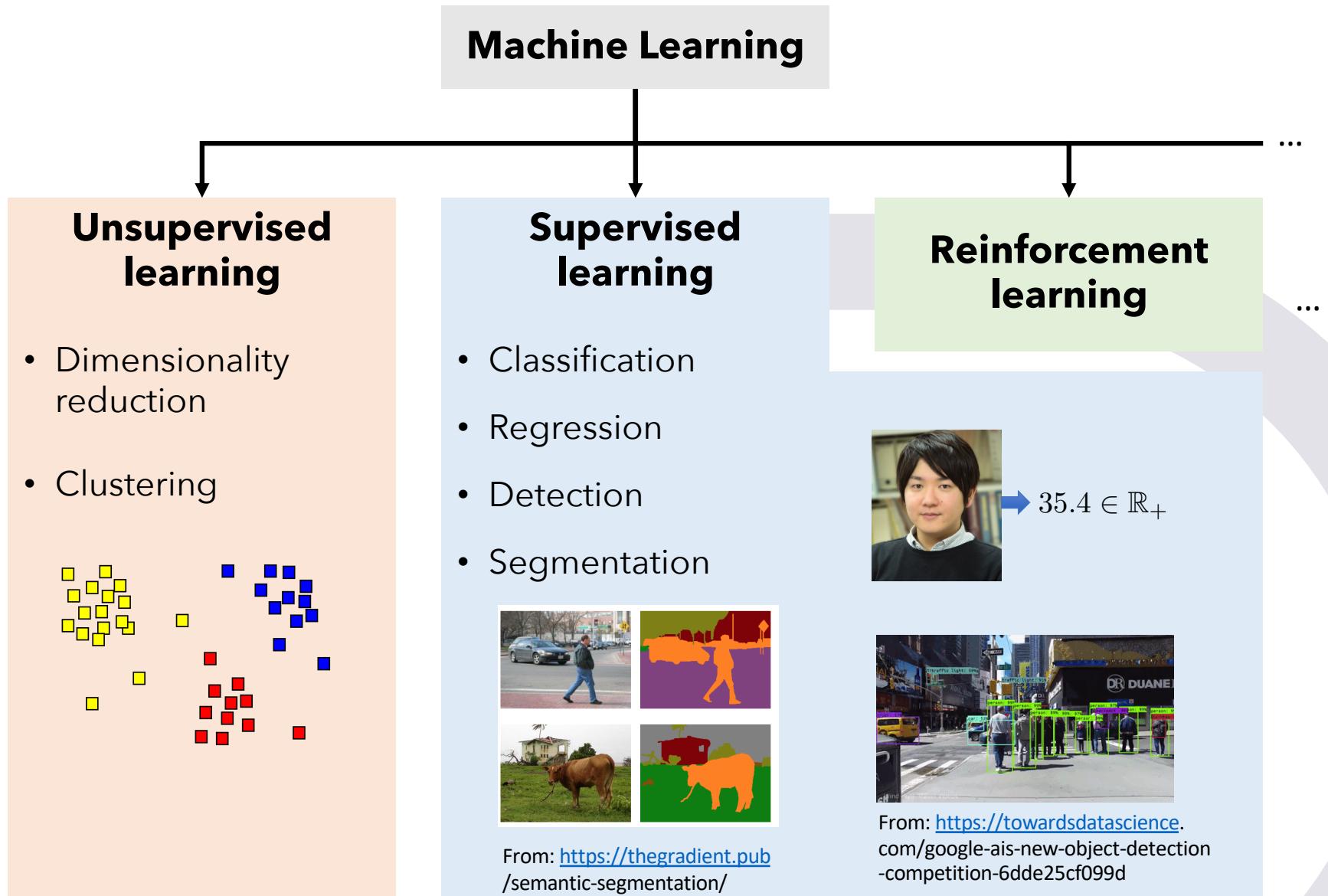
- (shallow) autoencoder
- Dictionary learning

Deep Learning, DL

- Restricted Boltzmann machine
- Deep neural networks

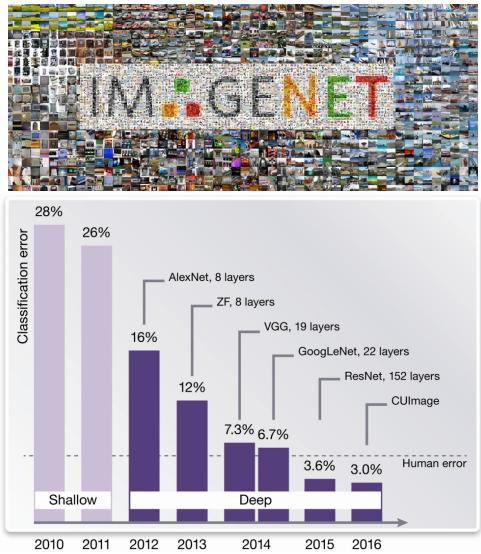


Tasks in Machine Learning

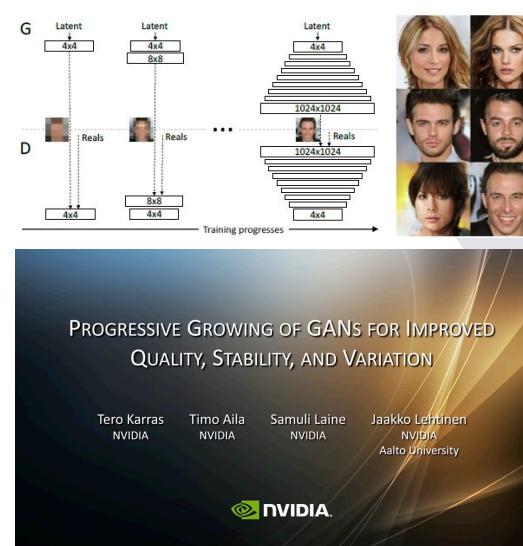


Why deep learning?

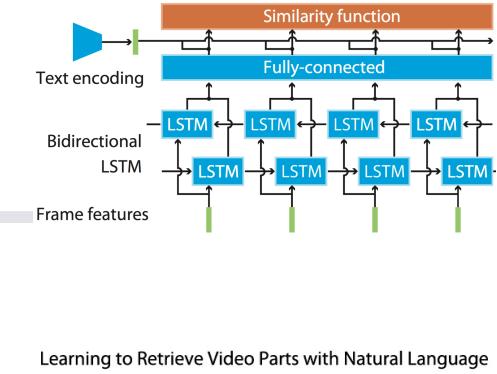
What deep learning can do



[Russakovsky 2015]

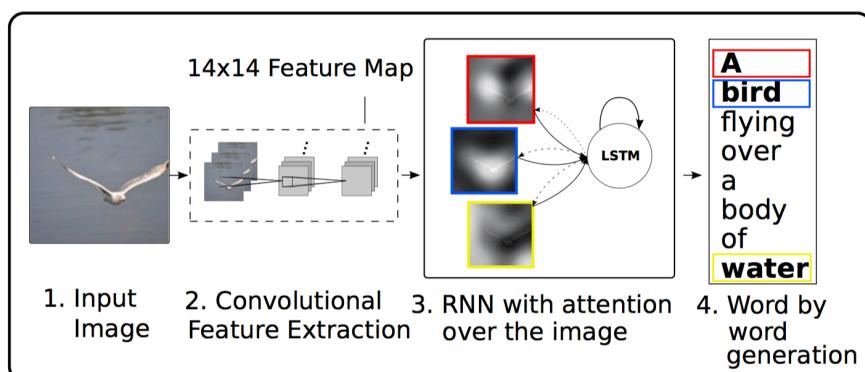


[Karras 2018]



Learning to Retrieve Video Parts with Natural Language

Example videos



[Xu 2015]

Visual to Sound: Generating Natural Sound
for Videos in the Wild

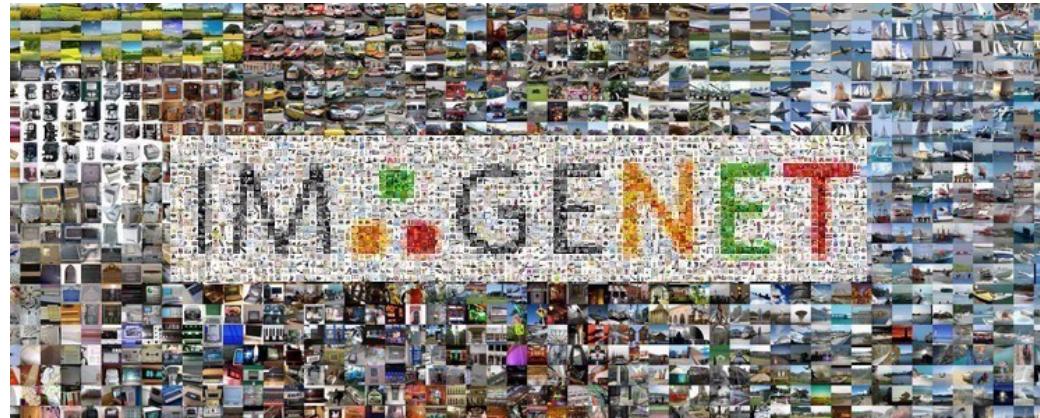
[Zhou 2017]



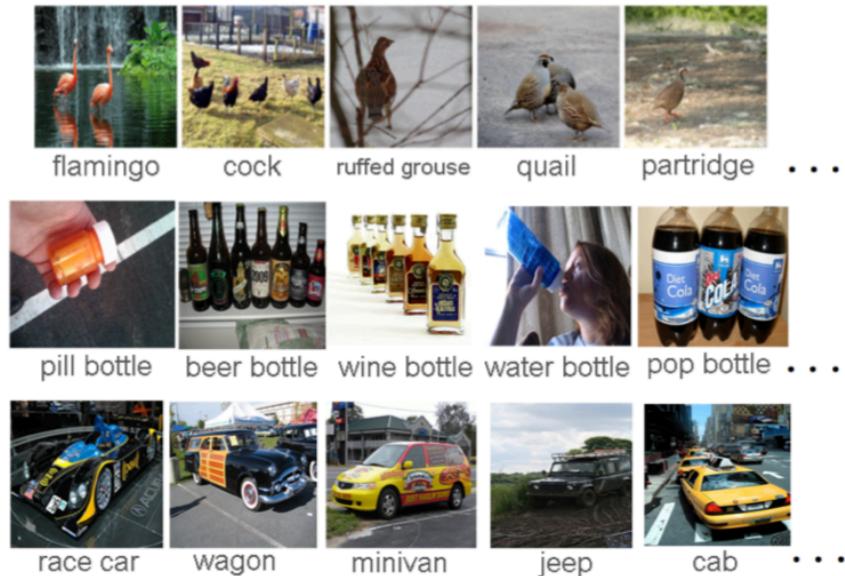
大阪大学データビリティフロンティア機構
Osaka University Institute for Datability Science

An image recognition task

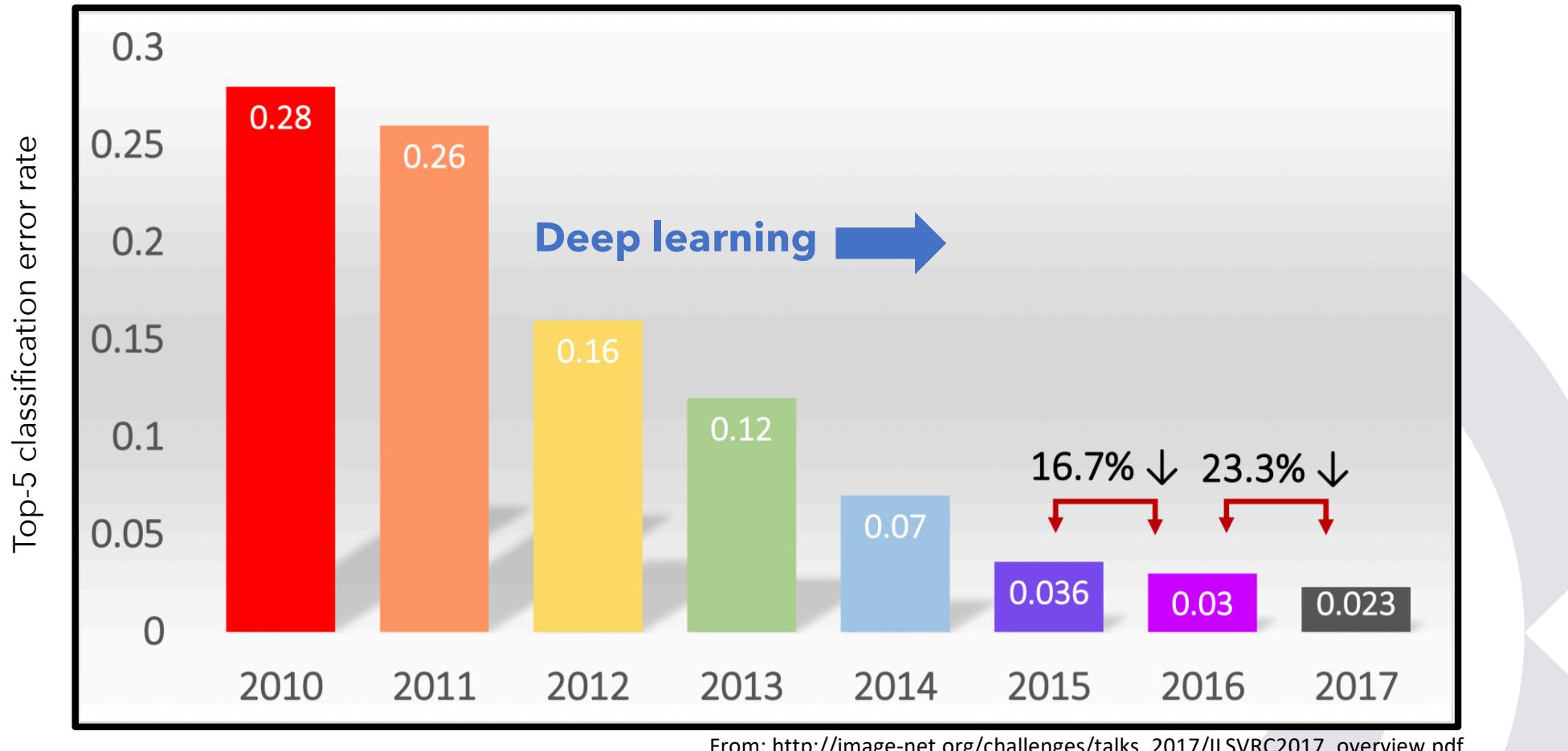
- ImageNet Large Scale Visual Recognition Competition



- 1,000 classes
- 1,461,406 images

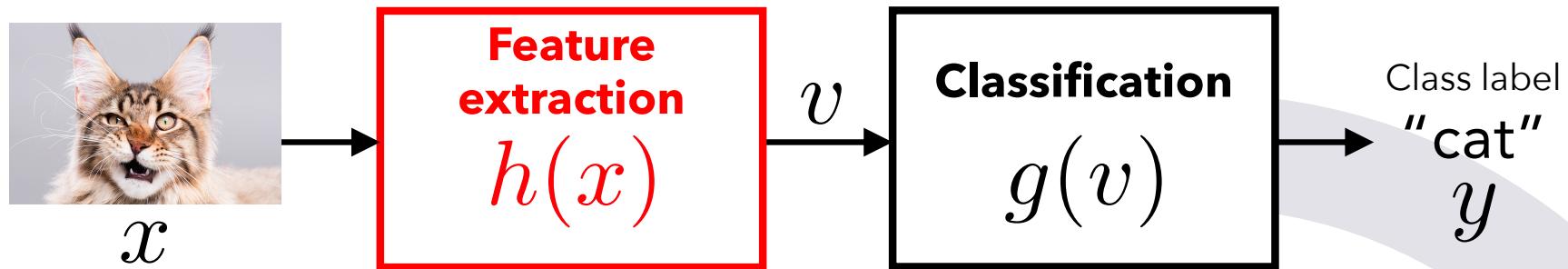


(Not so) recent models

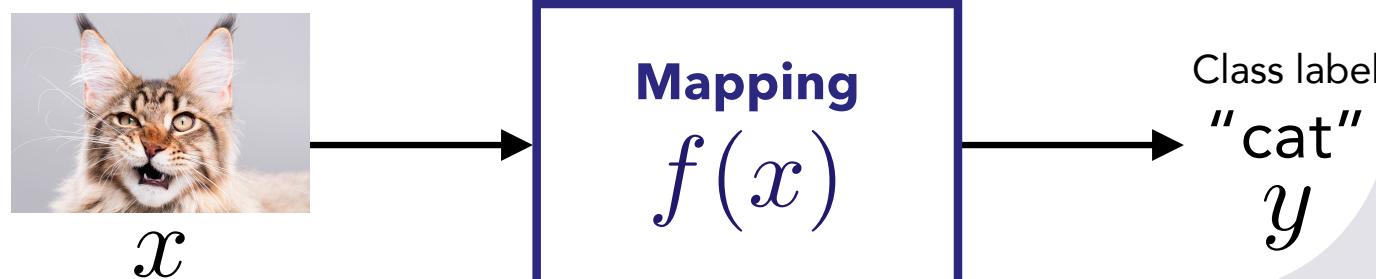


Difference between conventional machine learning and deep learning

- Conventional machine learning

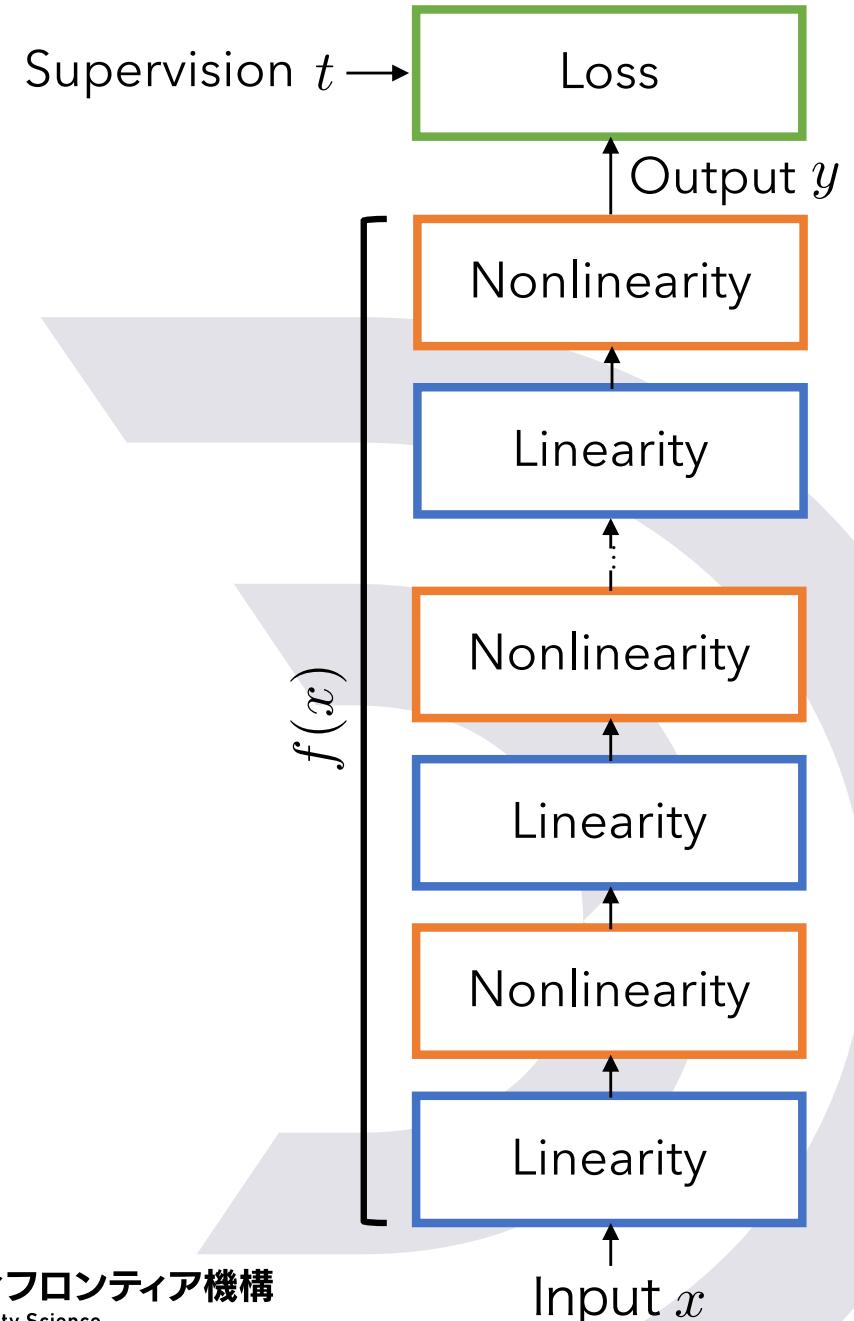


- Deep learning

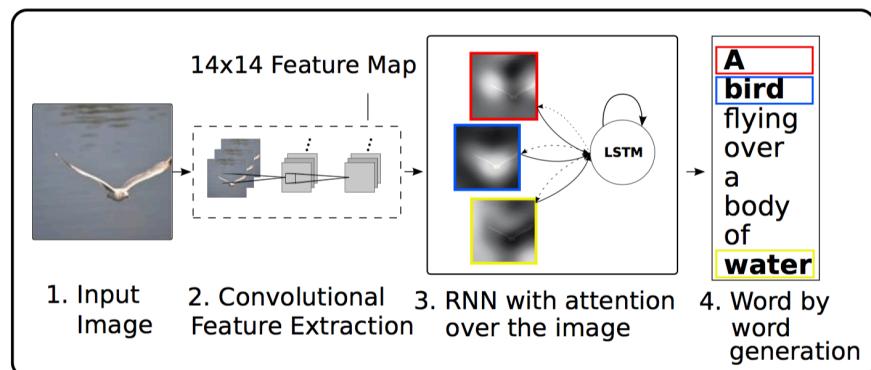
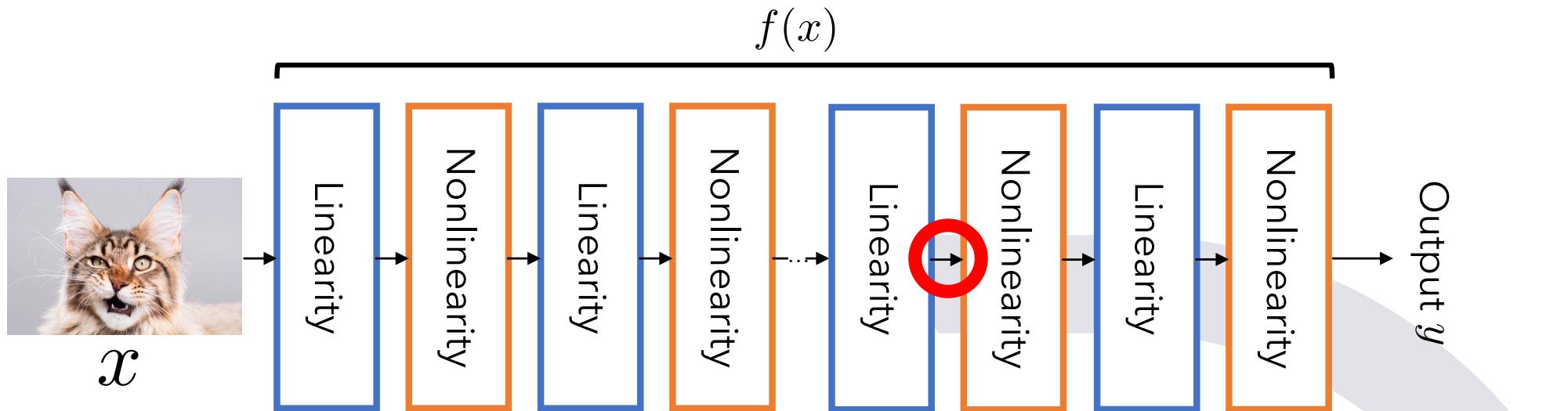


Building blocks of neural networks

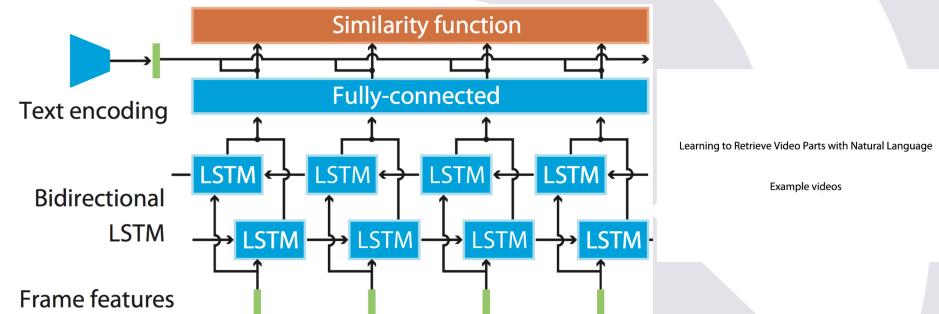
- **Linearity:** $h = Wx + b$
 - Convolution layer for images
- **Nonlinearity:** $y = \sigma(h)$
 - Hyperbolic tangent $\sigma(h) = \tanh(h)$
 - ReLU $\sigma(h) = \max(0, h)$
 - etc.
- **Loss:** $L(x, t)$
- **Others:**
 - Batch normalization
 - Poolong
 - etc.



What makes neural networks attractive?



[Xu 2015]



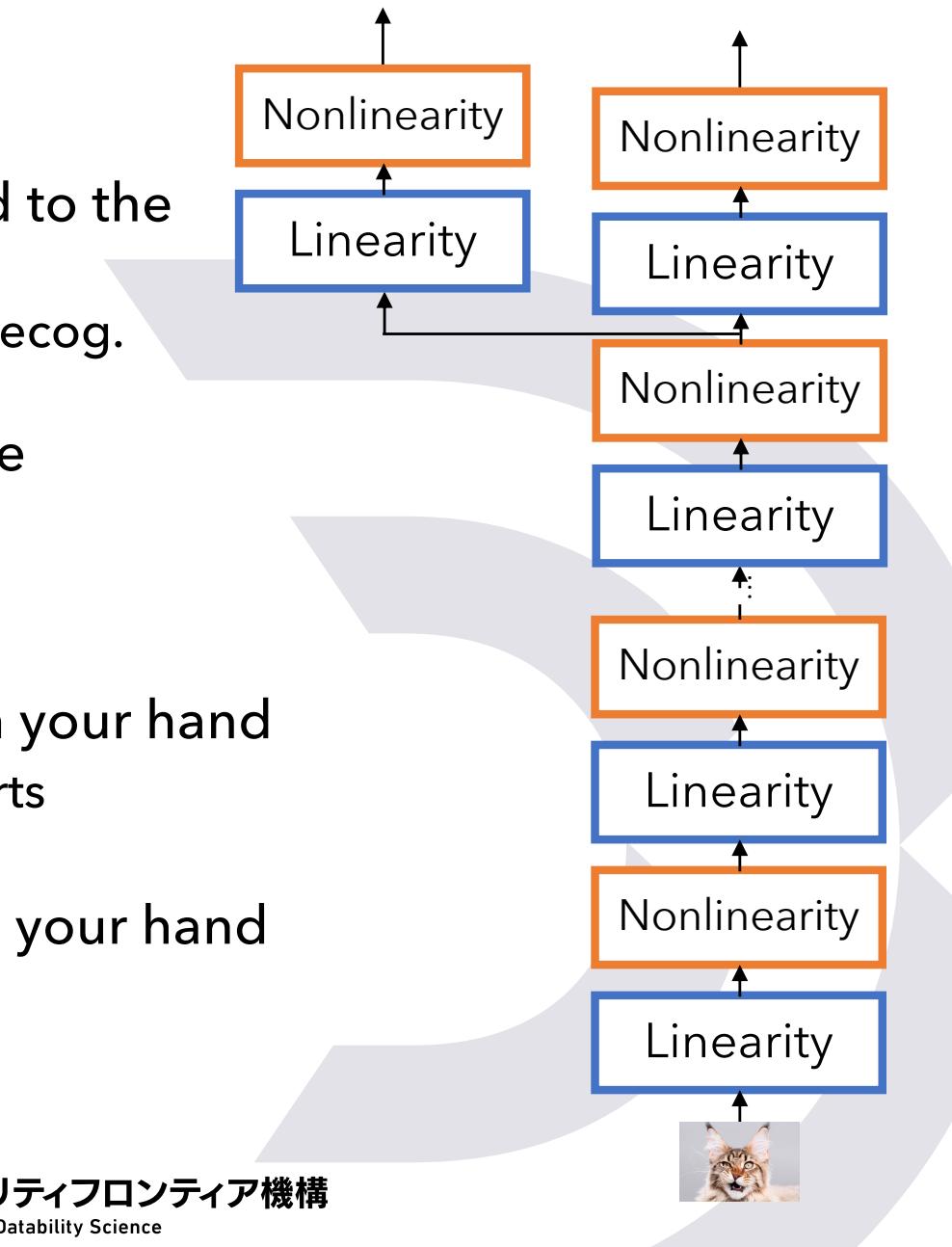
Pretraining and finetuning

• Pretraining

- Train your net with a task related to the task in your hand
 - E.g.) Train a net with an image recog. task and use it for captioning
- Larger datasets may be available

• Finetuning

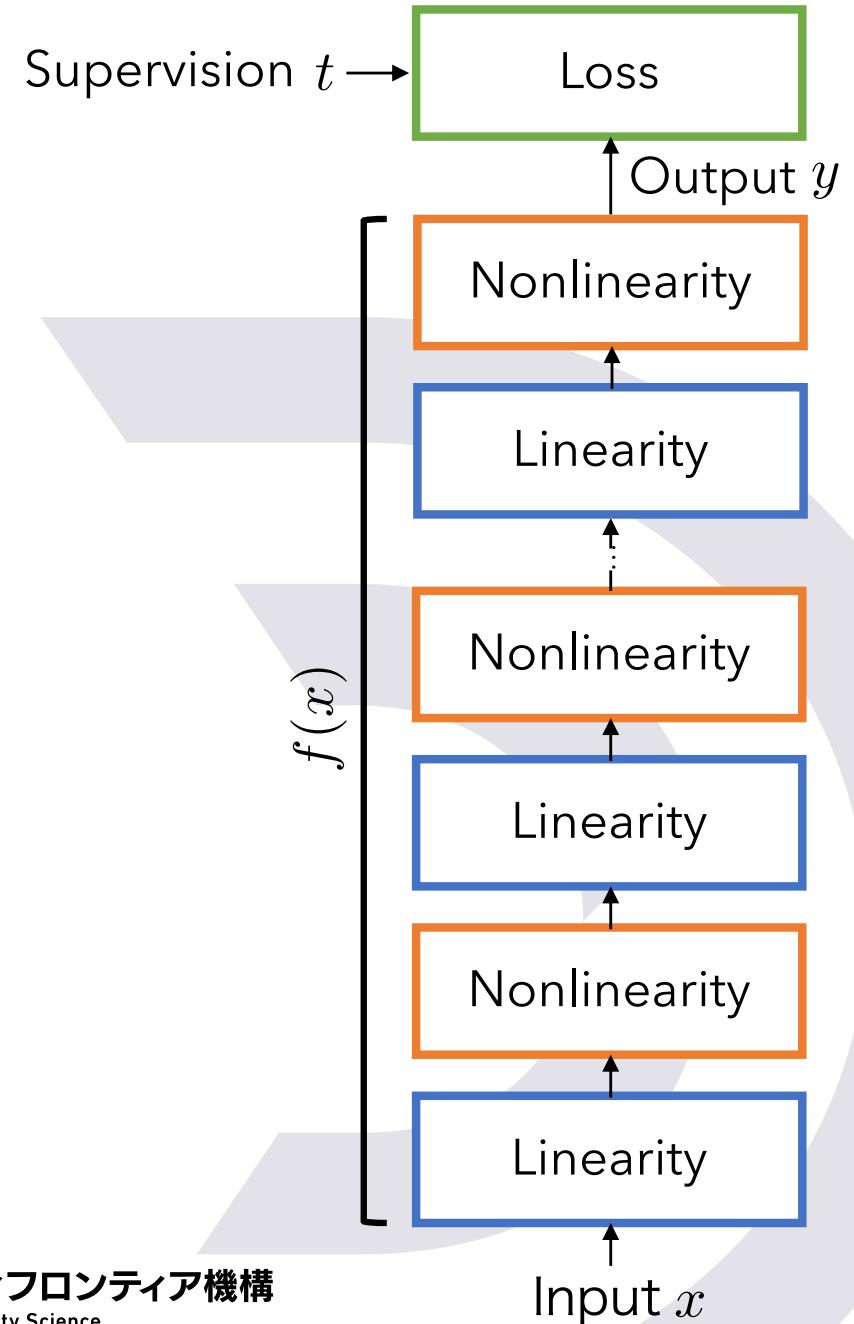
- Re-train your net with the task in your hand
 - You may train only replaced parts
 - Or you may train the entire net
- May work even when the task in your hand only has a small dataset



Training neural networks

Building blocks of neural networks (again)

- **Linearity:** $h = Wx + b$
 - Convolution layer for images
- **Nonlinearity:** $y = \sigma(h)$
 - Hyperbolic tangent $\sigma(h) = \tanh(h)$
 - ReLU $\sigma(h) = \max(0, h)$
 - etc.
- **Loss:** $L(x, t)$
- **Others:**
 - Batch normalization
 - Poolong
 - etc.



How to train your neural network

- Stochastic gradient descent
 - A type of gradient descent
- Back-propagation
 - The chain rule of differentiation;
differentiation for composite functions



Can you solve? (1)

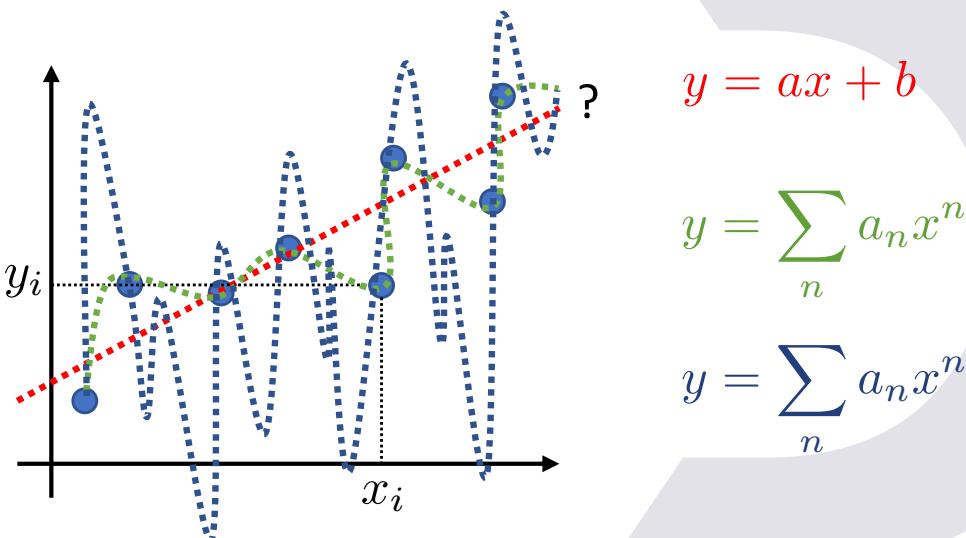
- Find a line that well explains the given data

$$D = \{(x_i, y_i) | i = 1, \dots, N\}$$

- What type of a line? **Neural network structure**
- With what criterion? **Selection of the loss**
- (How accurate fitting should be? **Regularization**)
- How to find it? **Selection of optimization algorithm**

Common choices:

- A straight line
- Mean squared error



Can you solve? (1)

- Suppose $f(x) = ax + b$ and MSE as the loss function

$$L(a, b) = \sum_i \|y_i - f(x_i)\|^2 = \sum_i (y_i - ax_i - b)^2$$

- What we need to find a and b that minimize the loss

$$\frac{\partial L}{\partial a} = 0$$



$$a^* = \frac{\frac{1}{N} \sum_i x_i y_i - (\frac{1}{N} \sum_i x_i)(\frac{1}{N} \sum_i y_i)}{\frac{1}{N} \sum_i x_i^2 - (\frac{1}{N} \sum_i x_i)^2} = \frac{\sigma_{xy}^2}{\sigma_{xx}^2}$$

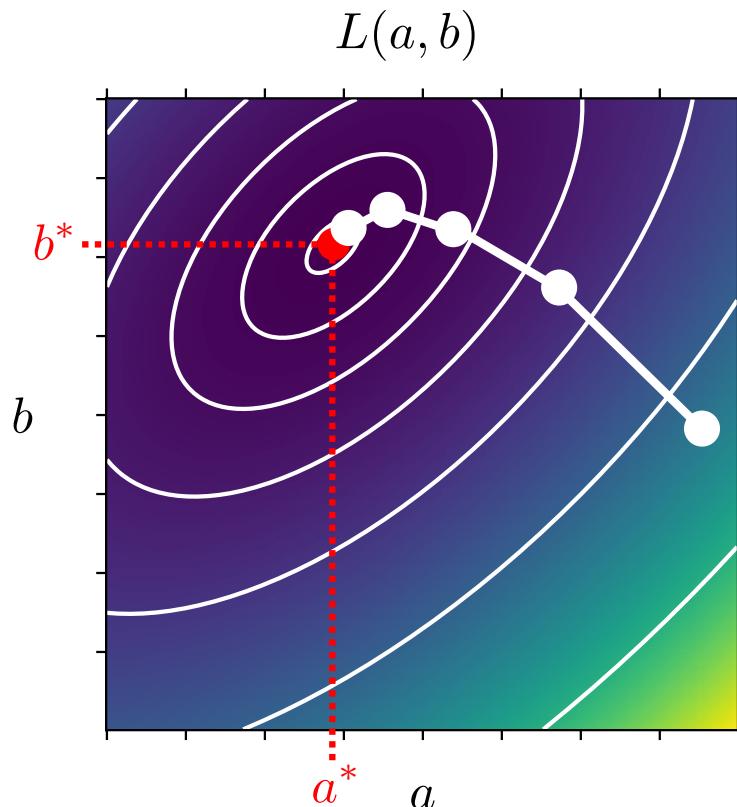
$$\frac{\partial L}{\partial b} = 0$$

$$b^* = \frac{1}{N} \sum_i y_i - a^* \frac{1}{N} \sum_i x_i = \mu_y - a^* \mu_x$$

There is an analytic solution!



Gradient Descent



- Find parameters with 0 gradients
- That is, shift the param. toward the opposite direction of the gradient from an initial point

$$a \leftarrow a - \alpha \frac{\partial L}{\partial a} \quad b \leftarrow b - \alpha \frac{\partial L}{\partial b}$$

Gradient descent

- In general: $\theta \leftarrow \theta - \alpha \frac{\partial L}{\partial \theta}$

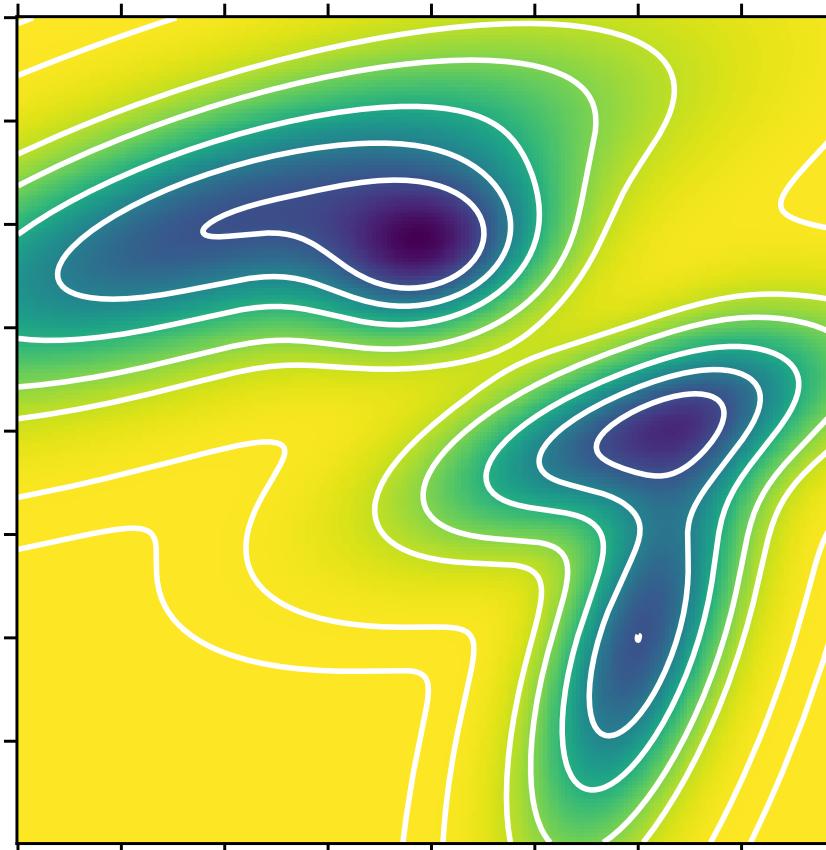
θ : parameters to learn

α : learning rate



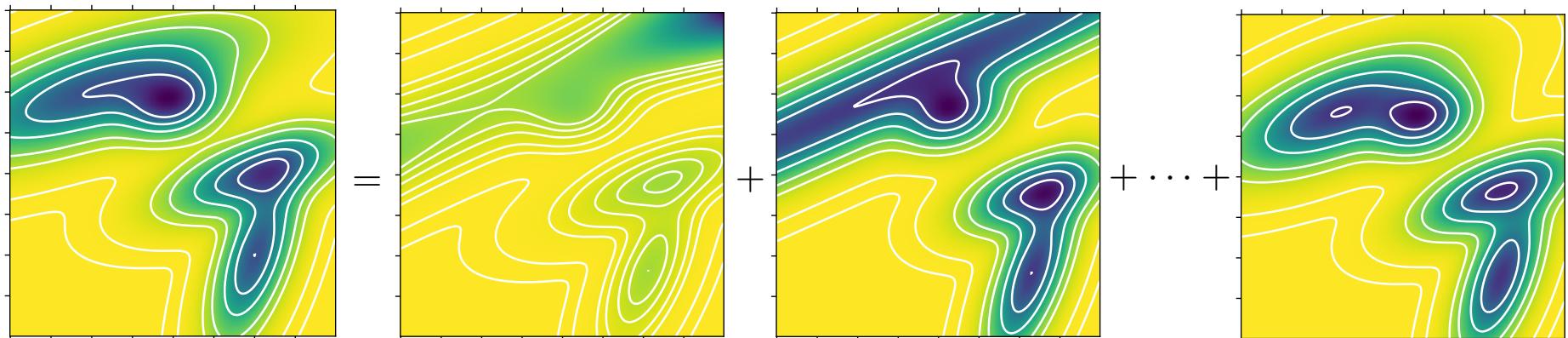
The neural network case

- Neural networks' loss function is multi-modal



Stochastic Gradient Descent

$$L(a, b) = \sum_i \|y_i - f(x_i)\|^2 = \frac{\sum_{i \in B_1} \|y_i - f(x_i)\|^2}{L_1(a, b)} + \frac{\sum_{i \in B_2} \|y_i - f(x_i)\|^2}{L_2(a, b)} + \cdots + \frac{\sum_{i \in B_N} \|y_i - f(x_i)\|^2}{L_N(a, b)}$$



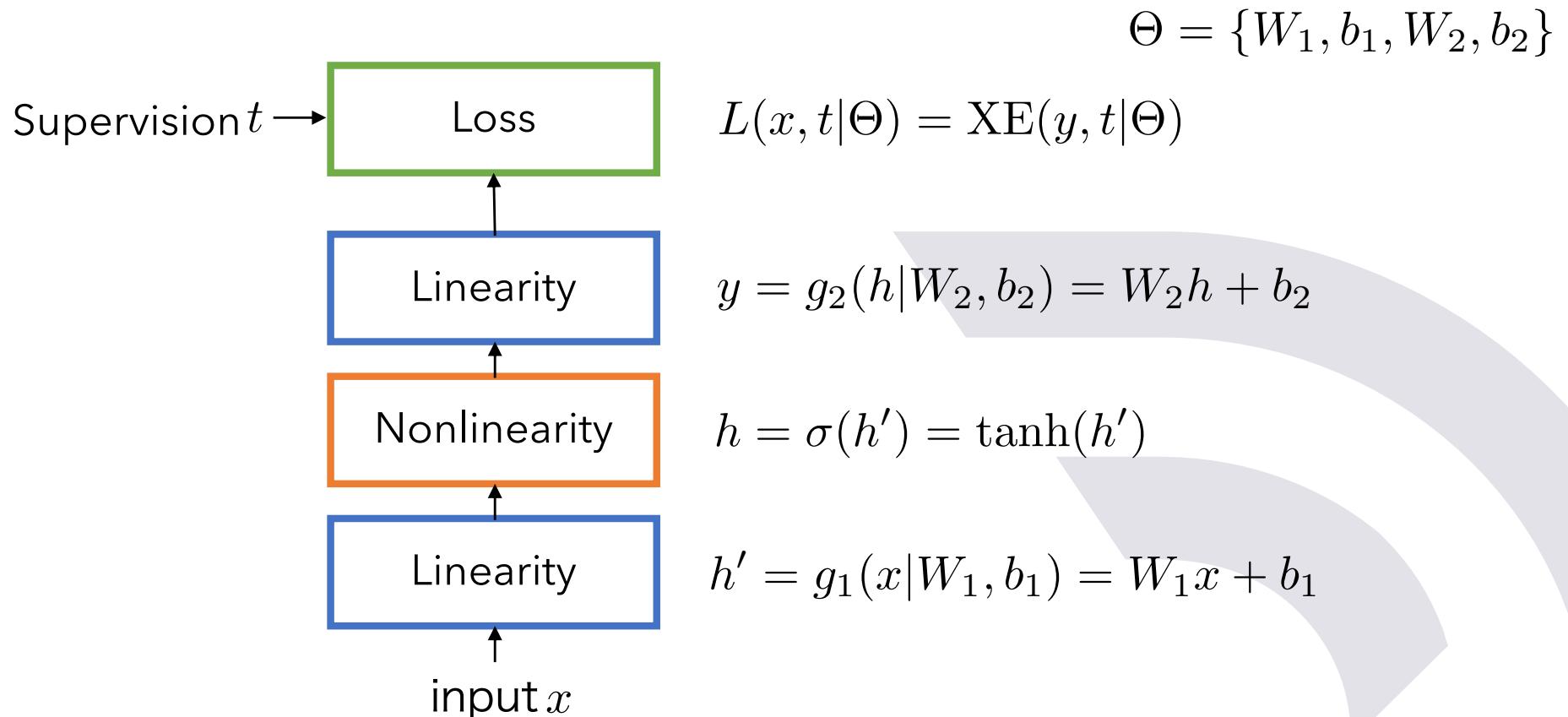
$$\theta \leftarrow \theta - \alpha' \frac{\partial L_n}{\partial \theta}$$

- Faster update
- Possibly go beyond local minima

Variants of stochastic gradient descent

- (Vanilla) stochastic gradient descent algorithm
- Stochastic gradient descent with momentum
- AdaGrad
- RMSProp
- Adam

Example: A simple neural network



Gradient required for SGD

$$\frac{\partial L}{\partial W_1} \quad \frac{\partial L}{\partial b_1} \quad \frac{\partial L}{\partial W_2} \quad \frac{\partial L}{\partial b_2}$$



Back-propagation, back-prop

- The chain rule: $\frac{\partial f(g(x))}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$

$$L(x, t|\Theta) = \text{XE}(y, t|\Theta) \implies \frac{\partial L}{\partial y}$$
$$y = g_2(h|W_2, b_2) = W_2 h + b_2 \implies \frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial W_2}, \quad \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial b_2}, \quad \frac{\partial L}{\partial h} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial h}$$
$$h = \sigma(h') = \tanh(h') \implies \frac{\partial L}{\partial h'} = \frac{\partial L}{\partial h} \frac{\partial h}{\partial h'}$$
$$h' = g_1(x|W_1, b_1) = W_1 x + b_1 \implies \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial h'} \frac{\partial h'}{\partial W_1}, \quad \frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial h'} \frac{\partial h'}{\partial b_1}, \quad \left(\frac{\partial L}{\partial x} = \frac{\partial L}{\partial h'} \frac{\partial h'}{\partial x} \right)$$



Problems in deep neural networks

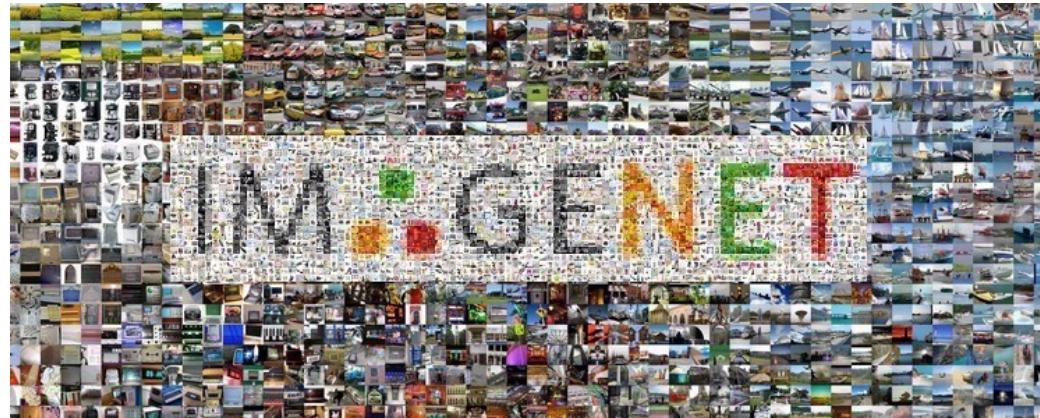
Universal approximation theorem says...

- Neural networks can approximate arbitrary (or a certain type of) functions
- Larger parameters give a better approximation
- A larger number of layers give a better approximation
- A FC layer (linearity) has MANY parameters
 - The number of parameters of a 1,000-D input and 1,000-D output layer is: **1,001,000**

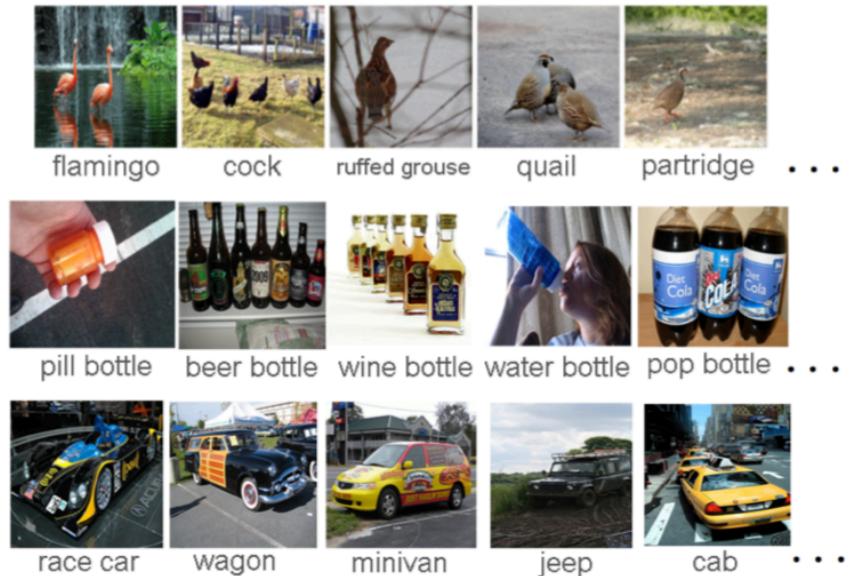


An image recognition task (again)

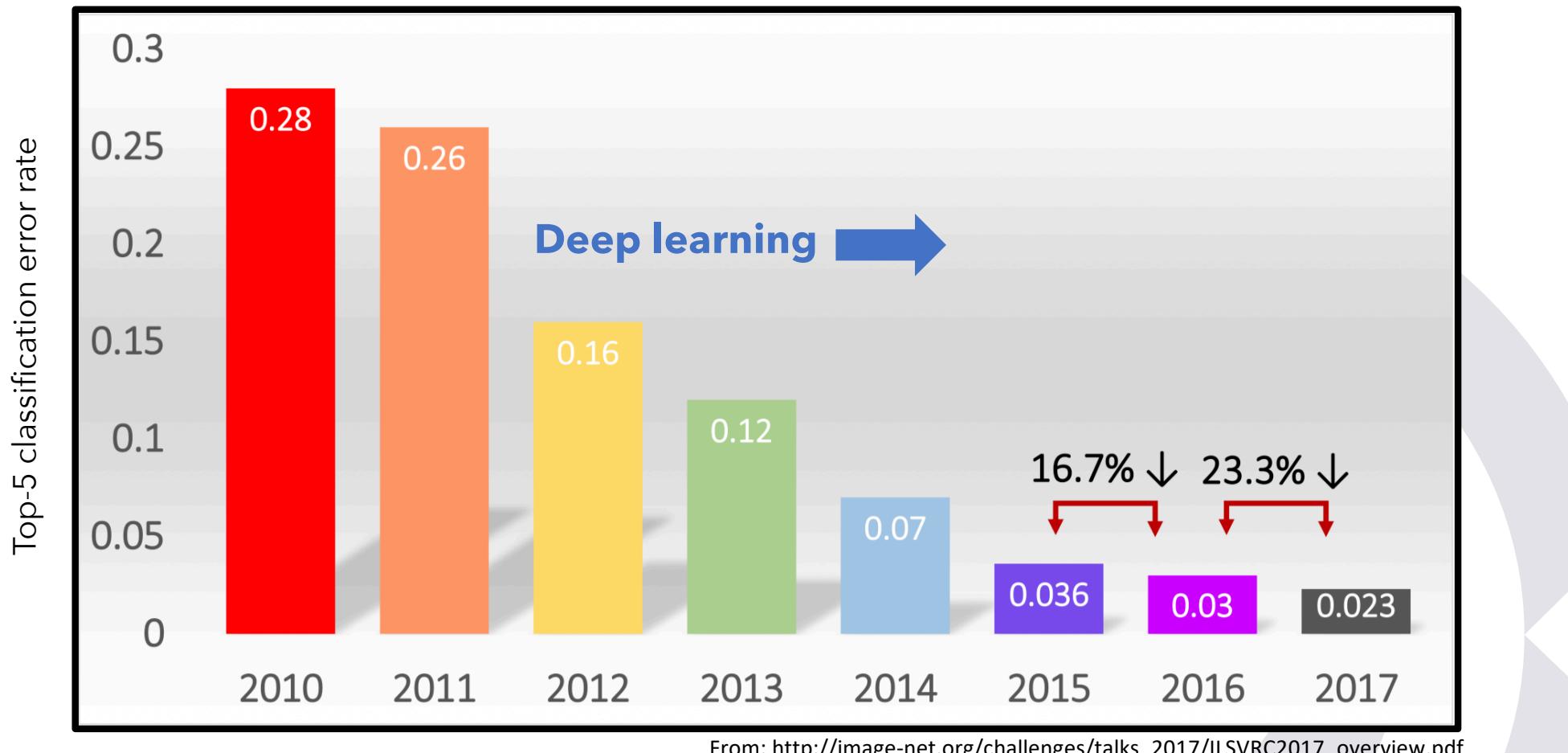
- ImageNet Large Scale Visual Recognition Competition



- 1,000 classes
- 1,461,406 images



(Not so) recent models (again)



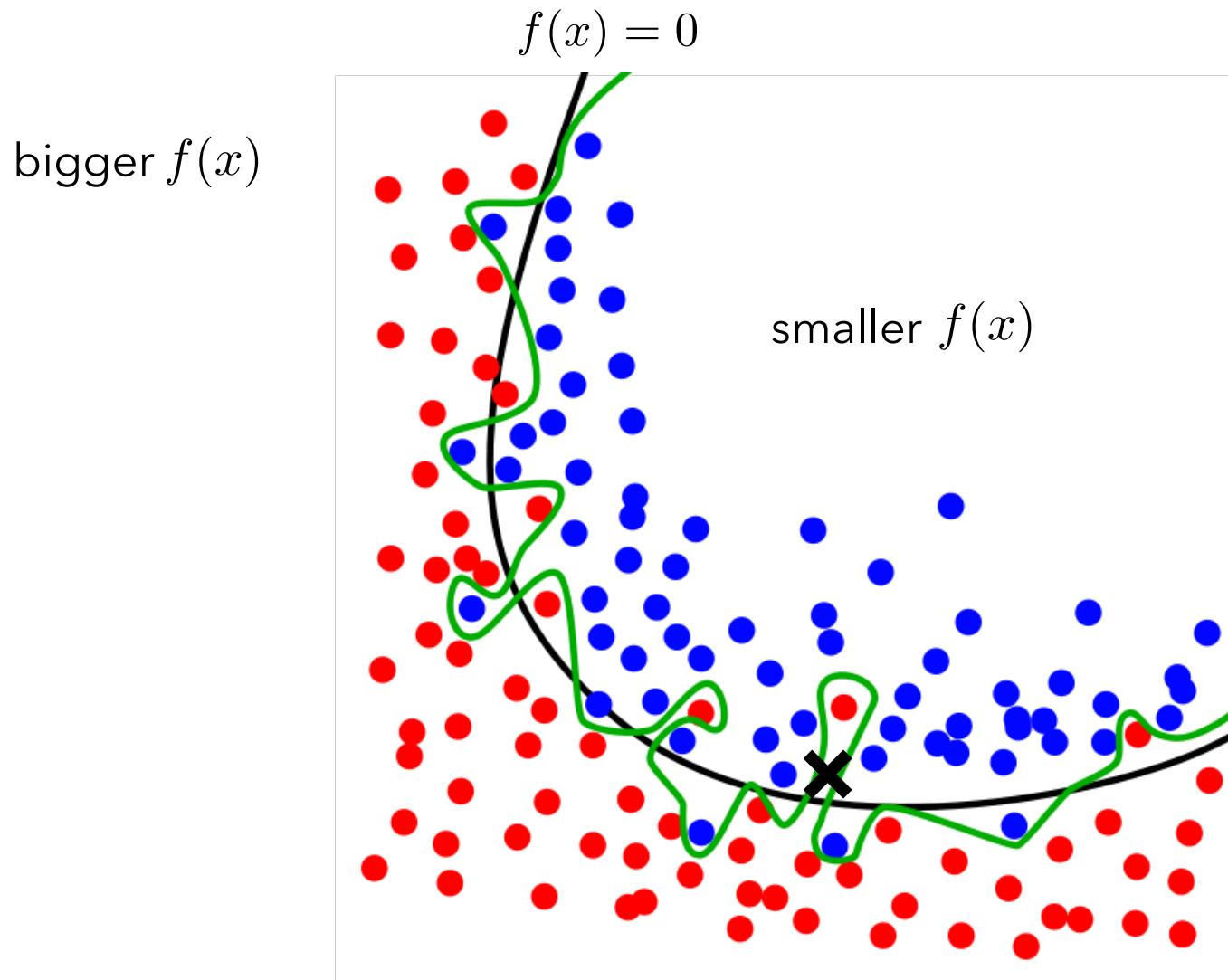
Comparison among models

	Performance	# layers	# Parameters
• AlexNet	0.16	8	61M
• VGG-19	0.088?	19	138M
• GoogLeNet	0.07	22	11M
• ResNet	0.036	152	25M

How does this happen?

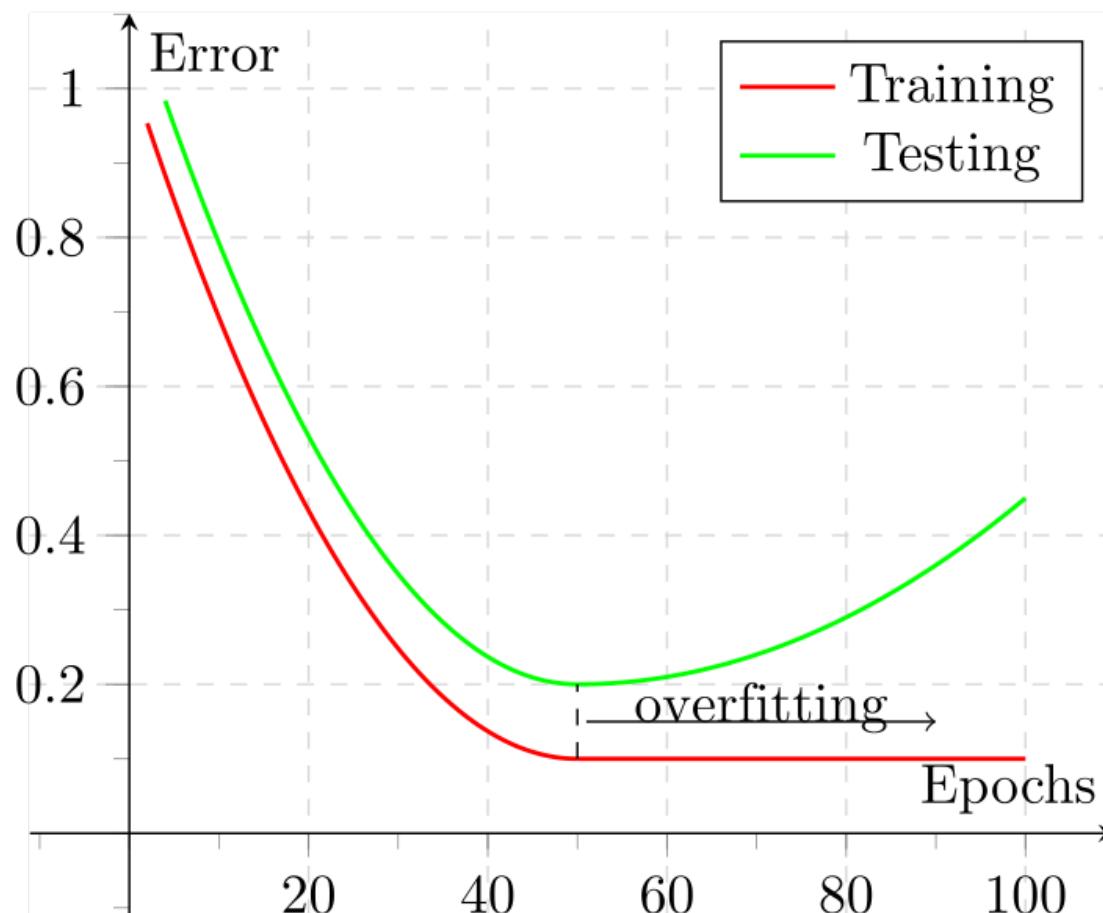


Which line is better?



Overfitting

- A model is too optimized to the training data and does not work well for unseen (test) data



What can we do?

- Convolution layers (for image inputs)
- Regularization
 - Dropout
 - Weight decay

Various problems

- Overfitting
 - Regularization (Dropout, weight decay)
 - Data augmentation
 - Larger amount of data
- Gradient vanishing
 - ReLU and its variants
 - Batch normalization
 - Skip-connection
 - Squeeze and excitation
- Hyperparameter tuning (# layers, learning rate, etc.)
 - Bayesian optimization
 - Researcher's experience

An application to Physics

Our collaboration with physics researchers

- Flavor classification

日本物理学会全国大会
Kishida, Iwasaki, et al.

- 3-class classification with variable-length data
 - Classification into cc / bb / uds events

- A classic approach hand-crafted features and use a shallow NN for classification

```
Axis = 1.336348e-01 1.270638e-01 9.828512e-01
3.600150e-01 2.364259e-02 2.897407e-01 1.600439e-01 1.355132e-04 -1.105776e-05 7.225226e-04
1.173989e+00 9.484937e-01 4.421300e-01 5.134615e-01 4.317461e-04 -9.262173e-04 -8.300396e-05
1.497720e+01 -2.482940e-01 -2.815872e+00 -1.470736e+01 6.750966e-04 -5.952772e-05 2.457871e-01
2.153059e+01 2.830333e-02 -5.172081e+00 -2.089666e-01 -1.711855e-04 -9.367832e-07 -8.197037e-03
1.474793e+00 -1.793663e-01 -2.634940e-01 -1.433286e+00 1.098547e-03 -7.478053e-04 -2.073109e-02
3.627837e+00 -4.301256e-01 -6.421792e-01 -3.541795e+00 -1.627690e-03 1.090212e-03 2.115826e-02
4.596739e+00 -2.664469e-01 -8.127258e-01 -4.514311e+00 7.059407e-02 -2.314381e-02 7.063290e-01
8.765116e+00 -2.147219e+00 3.149943e+00 -7.891456e+00 2.122391e-05 1.446769e-05 1.484219e-03
5.122196e-01 -1.974702e-01 7.846177e-02 -4.446685e-01 9.285856e-04 2.337036e-03 7.004034e-03
1.291064e+00 -2.293424e-01 2.425964e-01 -5.590074e-01 -3.947298e-03 5.869276e-05 -1.693335e-05 -9.371412e-04
5.983341e-01 -1.612783e-01 -5.590074e-01 -3.947298e-03 5.869276e-05 -1.693335e-05 -9.371412e-04
4.532778e-01 2.528231e-01 1.698745e-01 -3.052798e-01 3.536852e-04 -5.263873e-04 1.886640e-04
2.505493e+00 2.134482e+00 1.072338e+02 3.045616e+00 -2.055256e-06 4.090974e-04 -7.864043e-04
1.150009e+00 1.100459e+00 3.559504e-02 3.012568e-01 3.065370e-05 -9.476924e-04 -1.204631e-03

Axis = 5.468222e-01 8.021681e-01 -2.398164e-01
4.201890e+01 -2.260286e+01 -3.380862e+01 1.056676e+01 4.285707e-04 -2.865222e-04 -5.206644e-04
1.271398e+00 -9.386227e-01 -9.705339e-01 -3.016746e-01 -4.916799e-05 5.837850e-05 -1.463579e-04
7.126004e-01 -6.298689e-01 2.196849e-02 -3.018174e-01 4.816820e-05 1.381053e-03 -4.015306e-04
1.253496e+00 1.169279e+00 3.760114e-01 -2.077625e+00 8.086186e-04 -1.926894e-04 -4.435482e-05
3.165840e+00 6.937322e-01 2.911238e+00 -1.022977e+00 8.086186e-04 -1.926894e-04 -4.435482e-05
7.408828e+00 3.134073e+00 6.405586e+00 -2.004319e+00 9.545922e-04 -4.670551e-04 -3.828977e-05
8.365110e+00 4.553502e+00 6.939693e+00 -1.030462e+00 -2.500419e-04 1.640658e-04 -2.612930e-05
4.788787e+00 2.600081e+00 3.960575e+00 -6.829442e-01 -2.342055e-04 1.537537e-04 -5.630392e-04
3.193298e+01 -1.736678e+01 -2.564500e+01 7.773324e+00 -2.151546e-04 1.457026e-04 1.544318e-04
9.597038e+00 -4.968049e+00 -7.827353e+00 2.476827e+00 -7.555278e-05 4.795361e-05 -4.799157e-04
5.994379e-01 -2.606283e-01 -4.785615e-01 2.072592e-01 -4.846878e-04 2.639648e-04 2.408668e-04
1.509809e+01 -8.182952e+00 -1.203910e+01 4.004019e+00 -2.050355e-04 1.393622e-04 -1.444807e-03
5.880010e-01 -2.661611e-01 -3.555580e-01 3.591345e-01 6.112219e-04 -4.575149e-04 1.796477e-03
5.184984e-01 -3.343684e-01 -4.406086e-02 -3.682511e-01 -5.408319e-05 4.104257e-04 6.405641e-04
1.031485e+00 -8.673196e-02 -5.570330e-01 8.524468e-01 -3.491654e-03 5.436626e-04 1.576995e-03
5.767798e-01 4.807124e-01 8.317523e-02 -2.741975e-01 -3.317156e-03 1.917155e-04 3.945632e-04
7.069286e-01 1.592933e-01 6.699767e-01 7.755654e-02 -6.074407e-04 1.444250e-04 6.109483e-04
5.652046e-01 5.003294e-01 -9.607132e-02 7.025477e-04 -1.748612e-03 1.183762e-03
8.943885e+00 5.767168e+00 6.642895e+00 -1.607893e+00 -1.287538e-04 1.117803e+04 8.316932e-04
1.661733e+01 9.009043e+00 1.325890e+01 -4.378980e+00 -2.109194e-04 1.433561e-04 1.535002e-04
2.896062e+01 1.570820e+00 3.204081e+00 -7.814831e+00 9.634620e-06 -6.568455e-06 -2.311663e-04
8.448735e+01 -3.011931e-01 -7.731972e-01 7.597274e-02 -1.156947e+00 4.506800e-01 7.684941e-01
2.957734e+00 -1.846016e+00 -2.197302e+00 7.019877e-01 2.776428e-01 -2.332557e-01 -2.085605e-01

Axis = 5.974050e-01 1.278940e-01 -7.916757e-01
3.049313e-01 -1.623250e-01 -6.926404e-02 -2.057660e-01 -1.993575e-04 4.649383e-04 -2.028446e-03
7.209612e+00 1.938623e+00 -2.098141e+00 -6.618049e+00 -1.331579e-03 -1.230342e-03 6.710476e-04
5.150765e-01 1.658504e-01 -1.082958e-01 -4.545121e-01 -2.174730e-03 -3.330504e-03 7.520762e-03
1.678753e+00 -1.290594e+00 -2.777172e-01 -1.027601e+00 -2.396811e-05 1.113834e-04 -2.948495e-03
6.548846e-01 2.440873e-01 -6.591185e-02 5.877572e-01 -3.113475e-03 -4.856600e-03 -2.521130e-02
3.653267e+00 -2.359563e+00 -1.324441e+00 2.450477e-04 -4.433464e-04 7.898703e-04 -2.487400e-04
1.567742e+00 -1.162674e+00 -2.325280e+00 1.016094e+00 -2.251710e-04 1.125888e-03 -6.892465e-04
3.339994e+00 -2.530942e+00 5.964277e-02 2.174131e+00 -2.972407e+05 1.261342e-03 4.765579e-04
5.687092e+00 2.224050e+00 1.222246e+00 -5.087558e+00 2.174687e-04 -3.957153e-04 1.547311e-03
3.241751e-01 1.498072e-02 -8.894113e-02 -2.896002e-01 2.677136e-01 -1.032551e-01 -1.827803e+00
2.469065e+00 5.190706e-01 -4.482230e-01 -2.367795e-01 -2.130605e-03 -2.467376e-03 5.546636e-03
7.371763e-01 -5.912202e-01 -4.382983e-02 -4.153052e-04 1.485945e-04 3.346072e-04 -6.432181e-04
1.397669e+00 -7.329600e-01 2.139268e+01 1.162323e+00 -1.915298e-05 -6.562228e-05 2.637747e-03
1.323121e+00 -1.076112e+00 -4.241892e-01 6.270614e-01 -2.701272e-04 6.852772e-04 -6.450472e-04
1.405716e+00 -7.930601e-01 -5.207458e-01 1.027827e+00 4.824278e-04 -7.347045e-04 -7.288544e-04
1.346273e+01 -9.457000e+00 -3.075803e+00 9.073605e+00 -1.748960e-04 5.377430e-04 3.593510e-04
8.412551e+00 -5.518426e+00 -2.187031e+00 5.959479e+00 -5.581507e-05 1.408354e-04 1.103661e-03

Axis = 3.412007e-01 1.835908e-01 -9.218875e-01
6.053452e-01 3.034199e-01 3.451085e-01 -3.684948e-01 3.315940e-04 -2.915379e-04 9.809227e-05
2.242314e+01 4.252006e+00 2.555588e+00 -2.186704e+01 -6.775484e-04 1.127310e-03 1.631619e-03
4.615541e+00 8.030405e+00 4.904001e+00 -4.518591e+01 1.053577e-04 -1.725254e-04 -2.396392e-03
4.786486e+00 1.006292e+00 3.551062e-01 -4.663929e+00 5.546928e-04 -1.571977e-03 -2.141325e-02
2.076962e+00 -1.814143e+00 -9.196438e-01 3.967639e-01 -6.725764e-04 1.326763e-03 -9.279762e-04
2.990477e+00 -2.494993e+00 -1.465914e+00 7.413309e-01 1.613478e-04 -2.746147e-04 -5.321354e-04
5.352460e-01 3.369886e-01 -1.539671e-01 -3.601845e-01 8.891475e-04 1.946081e-03 -1.093706e-04
4.906971e-01 4.581336e-01 1.067404e-01 3.572935e-03 -1.206956e-04 5.180335e-04 6.616275e-04
1.510841e+00 4.635163e-01 1.346416e-01 -1.424844e+00 1.232350e-03 -4.242479e-03 -2.825476e-02
1.730957e+00 5.970972e-01 1.082349e-01 -1.615080e+00 6.187534e-04 -3.413463e-03 1.363038e-02
5.387886e+00 -2.463795e+00 -4.764452e+00 4.894180e-01 -6.337707e+01 3.277357e+01 -7.450118e-01
6.558535e+00 -6.543017e+00 -6.038268e+00 4.244774e+00 5.559014e-01 -6.023701e+01 3.993892e+00
6.793306e-01 1.701942e-01 2.753585e-03 -6.426727e-01 -7.009438e-03 4.332408e-01 -1.019863e+01
```



Power of deep learning?

- Look into the data...

```
-----  
Axis = 1.336348e-01 1.270638e-01 9.828512e-01  
3.600150e-01 2.364259e-02 2.897407e-01 1.600439e-01 1.355132e-04 -1.105776e-05 7.225226e-04  
1.173989e+00 9.484937e-01 4.421300e-01 5.134615e-01 4.317461e-04 -9.262173e-04 -8.300396e-05  
1.497720e+01 -2.482940e-01 -2.815872e+00 -1.470736e+01 6.750966e-04 -5.952772e-05 2.457871e-01  
2.153059e+01 2.830333e-02 -5.172081e+00 -2.089966e+01 -1.711855e-04 -9.367832e-07 -8.197037e-03  
1.474923e+00 -1.793663e-01 -2.634940e-01 -1.433286e+00 1.098547e-03 -7.478053e-04 -2.073109e-02  
3.627837e+00 -4.301256e-01 -6.421792e-01 -3.541795e+00 -1.627690e-03 1.090212e-03 2.115826e-02  
4.596739e+00 -2.664469e-01 -8.127258e-01 -4.514311e+00 7.059407e-02 -2.314381e-02 7.063290e-01  
8.765116e+00 -2.147219e+00 3.149943e+00 -7.891456e+00 2.122391e-05 1.446769e-05 1.484219e-03  
5.122196e-01 -1.974702e-01 7.846177e-02 -4.446685e-01 9.285856e-04 2.337036e-03 7.004034e-03  
1.291064e+00 -2.293424e-01 2.425964e-01 -1.239318e+00 -1.155399e-02 -1.092275e-02 1.093272e-02  
5.983341e-01 -1.612783e-01 -5.590074e-01 -3.947298e-03 5.869276e-05 -1.693335e-05 -9.371412e-04  
4.532778e-01 2.528231e-01 1.698745e-01 -3.052798e-01 3.536852e-04 -5.263873e-04 1.886640e-04  
2.505493e+00 2.134482e+00 1.072338e-02 1.304561e+00 -2.055256e-06 4.090974e-04 -7.864043e-04  
1.150009e+00 1.100459e+00 3.559504e-02 3.012568e-01 3.065370e-05 -9.476924e-04 -1.204631e-03  
-----  
Axis = 5.468222e-01 8.021681e-01 -2.398164e-01  
4.201890e+01 -2.260286e+01 -3.380862e+01 1.056676e+01 4.285707e-04 -2.865222e-04 -5.206644e-04  
1.271299e+00 -9.396227e-01 -7.805229e-01 -3.016746e-01 -4.016700e-05 5.927950e-05 -1.162579e-04
```

Okay, don't think, do

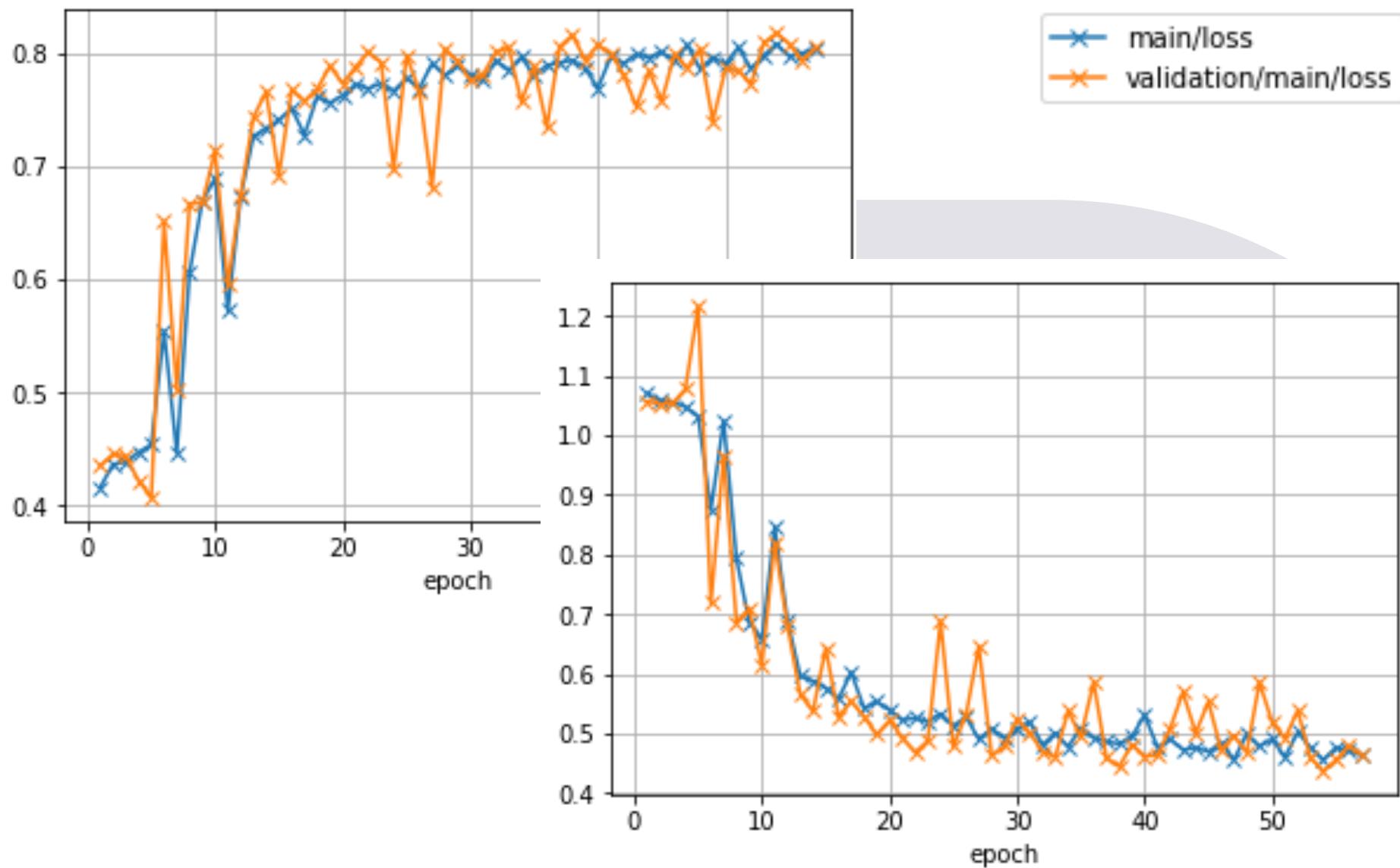


The code (200+ lines)



- Loading required modules
 - Reading data
 - Defining the neural network structure
 - Evaluation
 - Configuration and training

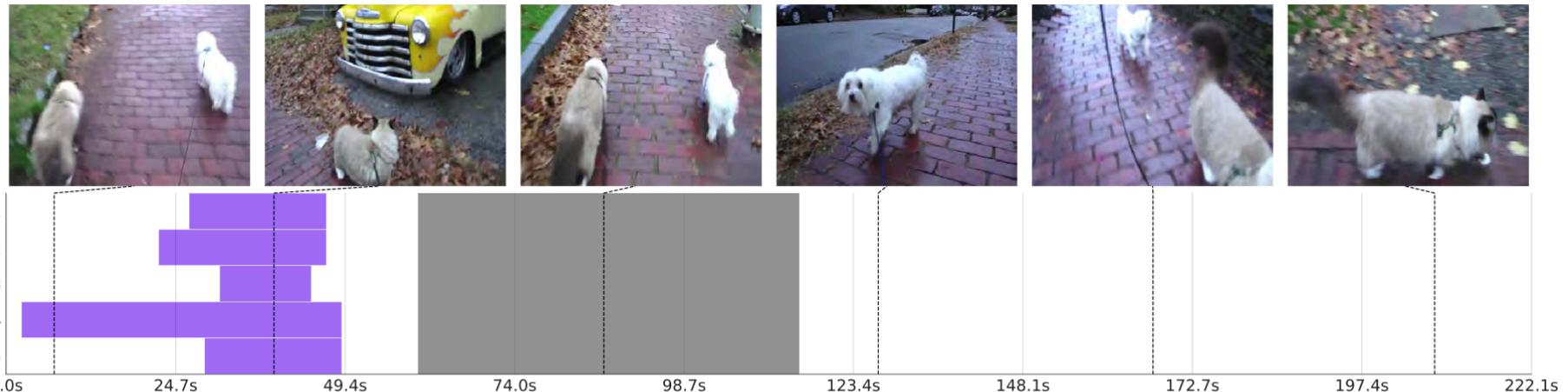
What we got



Recent tales of neural networks

Lazy deep neural networks?

Video Moment Retrieval



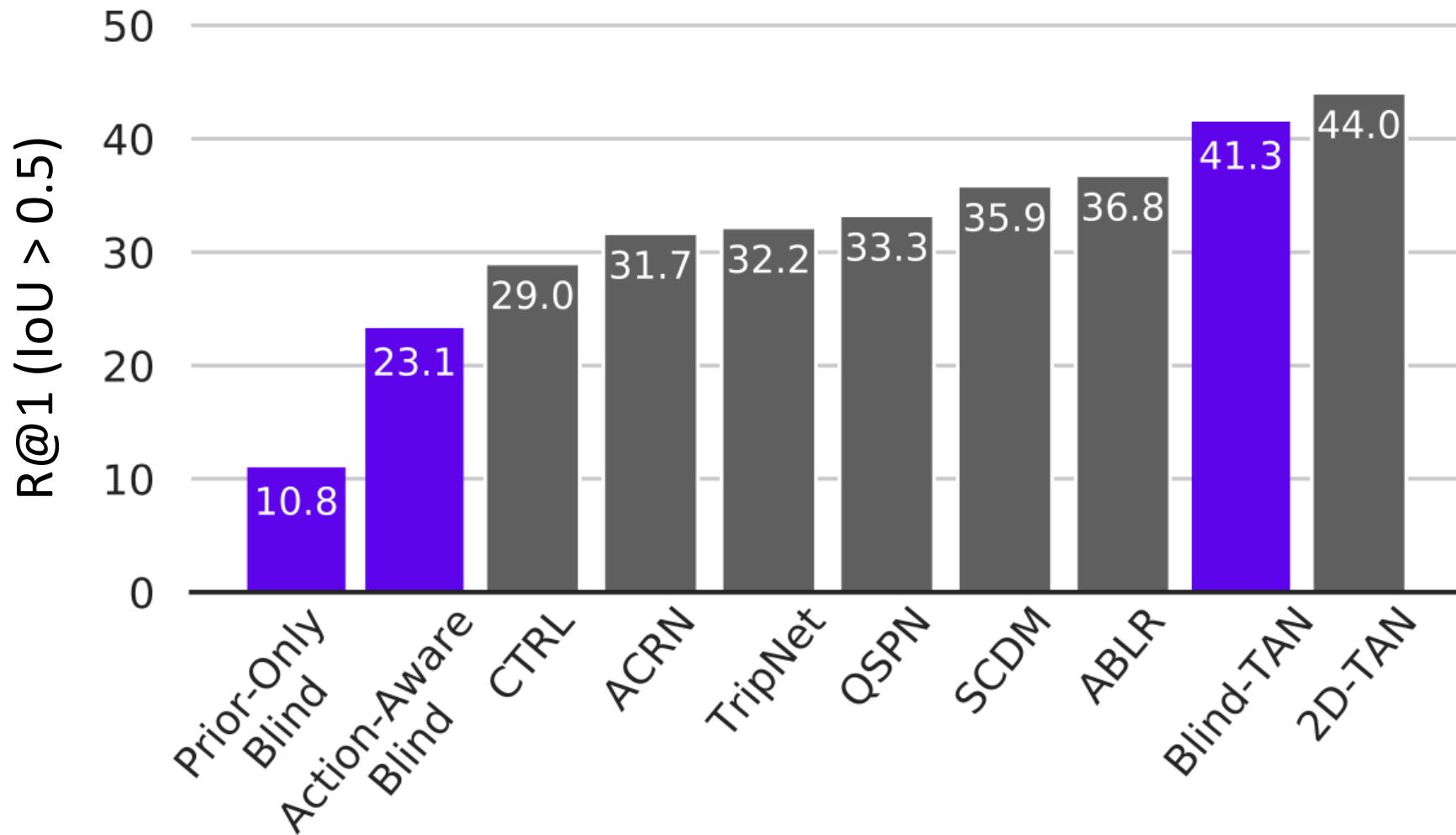
As the walk continues, the cat stops and begins staring at a parked car with large red flames painted on the side.



A crowd of people are cheering.

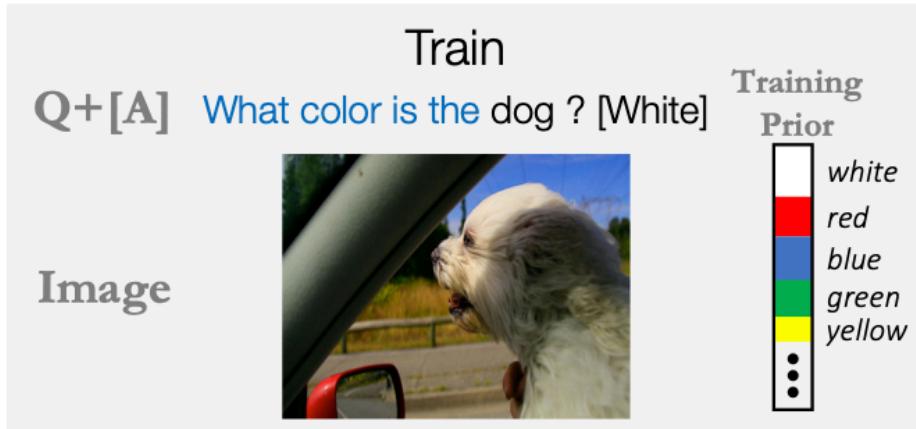


On a certain dataset



Visual Question Answering

Example 1



Example 2



Comparison

Model	Dataset	Overall	Yes/No	Number	Other	Dataset	Overall	Yes/No	Number	Other
per Q-type prior [5]	VQA v1	35.13	71.31	31.93	08.86	VQA v2	32.06	64.42	26.95	08.76
	VQA-CP v1	08.39	14.70	08.34	02.14	VQA-CP v2	08.76	19.36	11.70	02.39
d-LSTM Q [5]	VQA v1	48.23	79.05	33.70	28.81	VQA v2	43.01	67.95	30.97	27.20
	VQA-CP v1	20.16	35.72	11.07	08.34	VQA-CP v2	15.95	35.09	11.63	07.11
d-LSTM Q + norm I [24]	VQA v1	54.40	79.82	33.87	40.54	VQA v2	51.61	73.06	34.41	39.85
	VQA-CP v1	23.51	34.53	11.40	17.42	VQA-CP v2	19.73	34.25	11.39	14.41
NMN [3]	VQA v1	54.83	80.39	33.45	41.07	VQA v2	51.62	73.38	33.23	39.93
	VQA-CP v1	29.64	38.85	11.23	27.88	VQA-CP v2	27.47	38.94	11.92	25.72
SAN [39]	VQA v1	55.86	78.54	33.46	44.51	VQA v2	52.02	68.89	34.55	43.80
	VQA-CP v1	26.88	35.34	11.34	24.70	VQA-CP v2	24.96	38.35	11.14	21.74
MCB [11]	VQA v1	60.97	81.62	34.56	52.16	VQA v2	59.71	77.91	37.47	51.76
	VQA-CP v1	34.39	37.96	11.80	39.90	VQA-CP v2	36.33	41.01	11.96	40.57

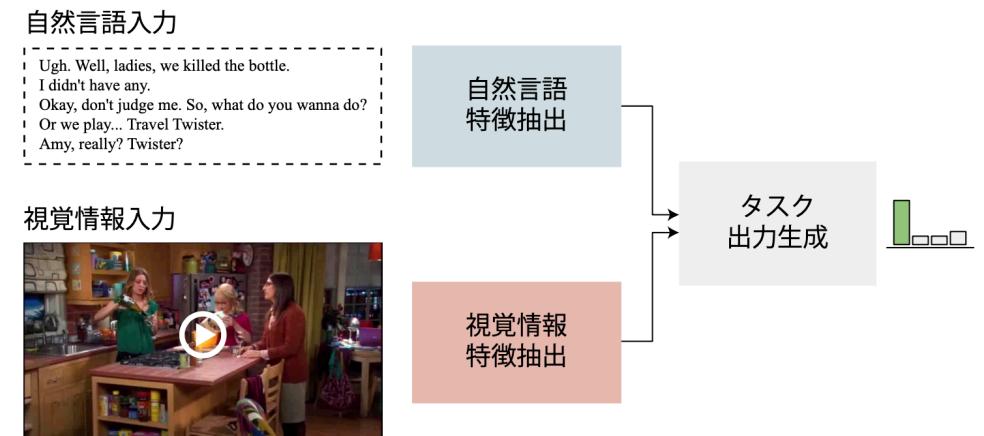
- Datasets with "-CP" reduce dataset bias
- VQA v1/v2 can be answered with 50% accuracy based only on questions
- Even SoTA models performed < 40% accuracy over "-CP" datasets



So...?

**Vision hardly helps
for vision + language (=multimodal) tasks**

- Deep neural networks are lazy?
 - Shortcut learning
[Geirhos et al., arXiv:2004:077803]
- They are trapped in superficial correlations
 - “There could be better ways...”
 - “But it’s working...” (though it’s a local minimum)
- **Note this when you use multiple different inputs**



Takeaways

- Deep learning is a part of machine learning
- Very easy and handy
 - No hand-crafted features
 - Neural networks are mostly a stack of multiplications and additions
- But don't forget the puzzling problems
 - Overfitting
 - Vanishing gradients
 - Hyperparameter tuning
 - Shortcut learning?





大阪大学データビリティフロンティア機構
Osaka University Institute for Datability Science