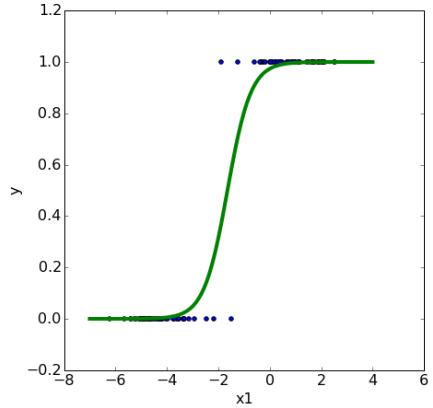
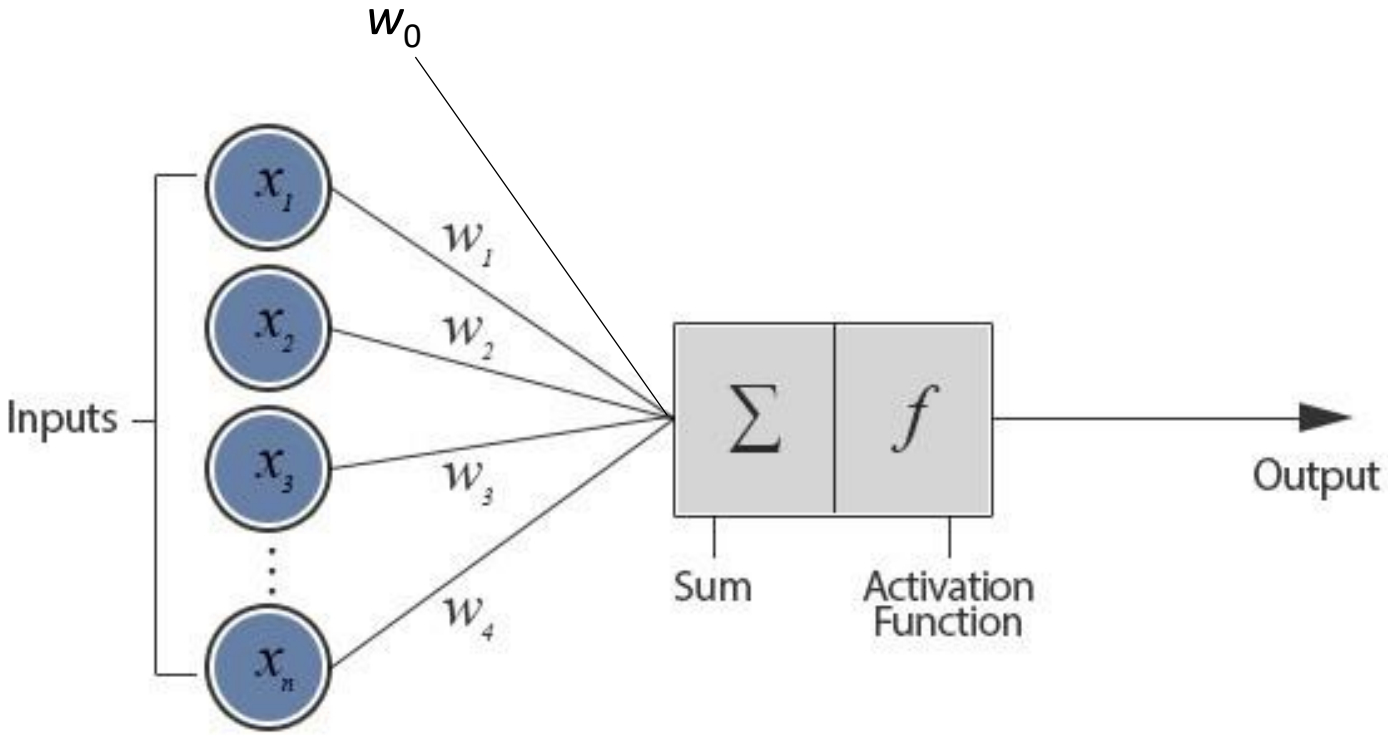
$$f(x,\theta) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_m x_m) = g(\theta' x)$$





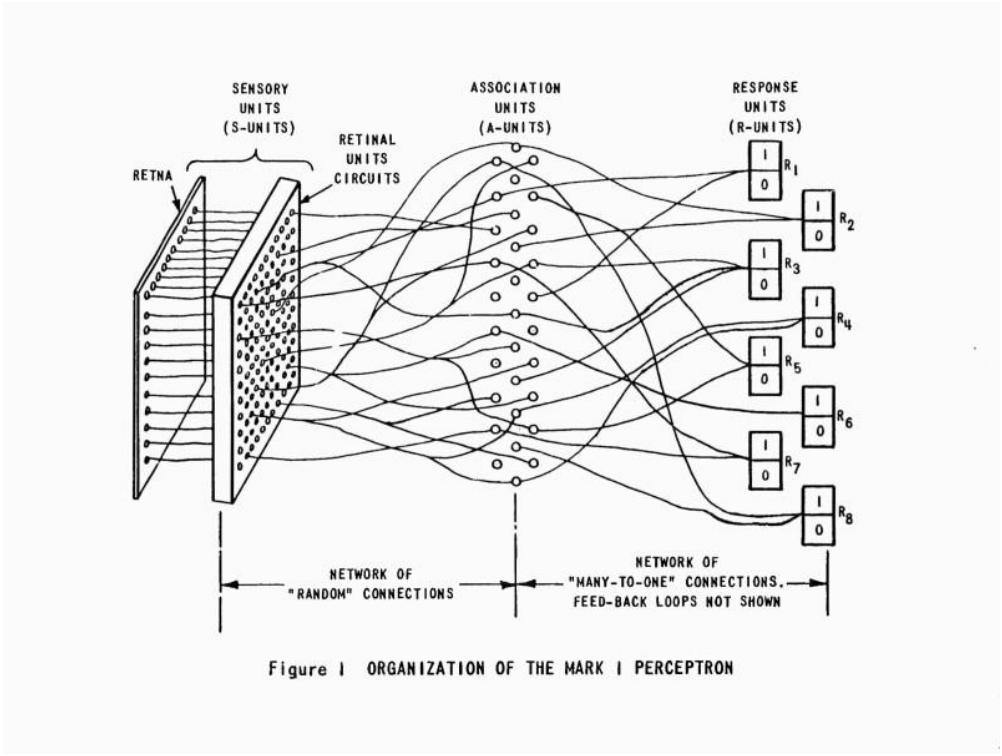


Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

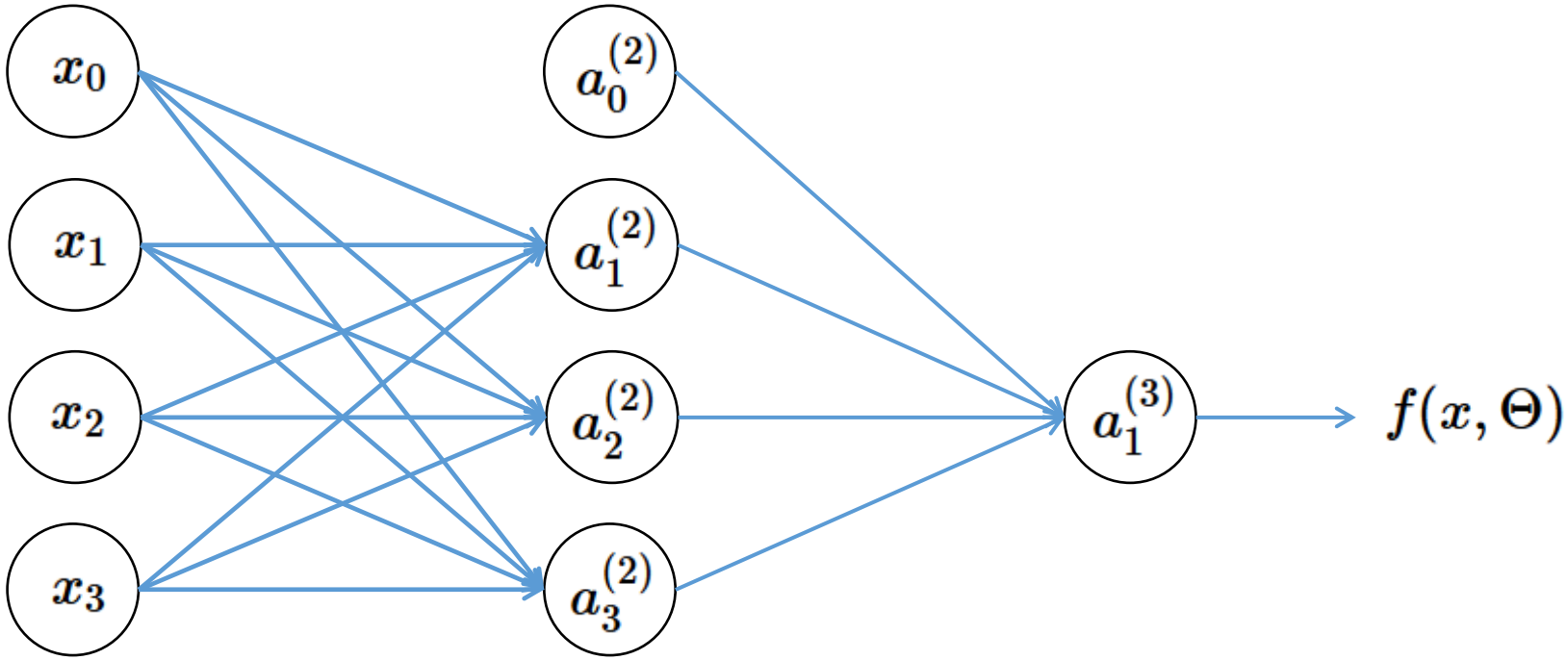
Frank Rosenblatt, Mark I Perceptron Operators' Manual. Buffalo, NY: Cornell Aeronautical Laboratory, 1960

Model: $f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

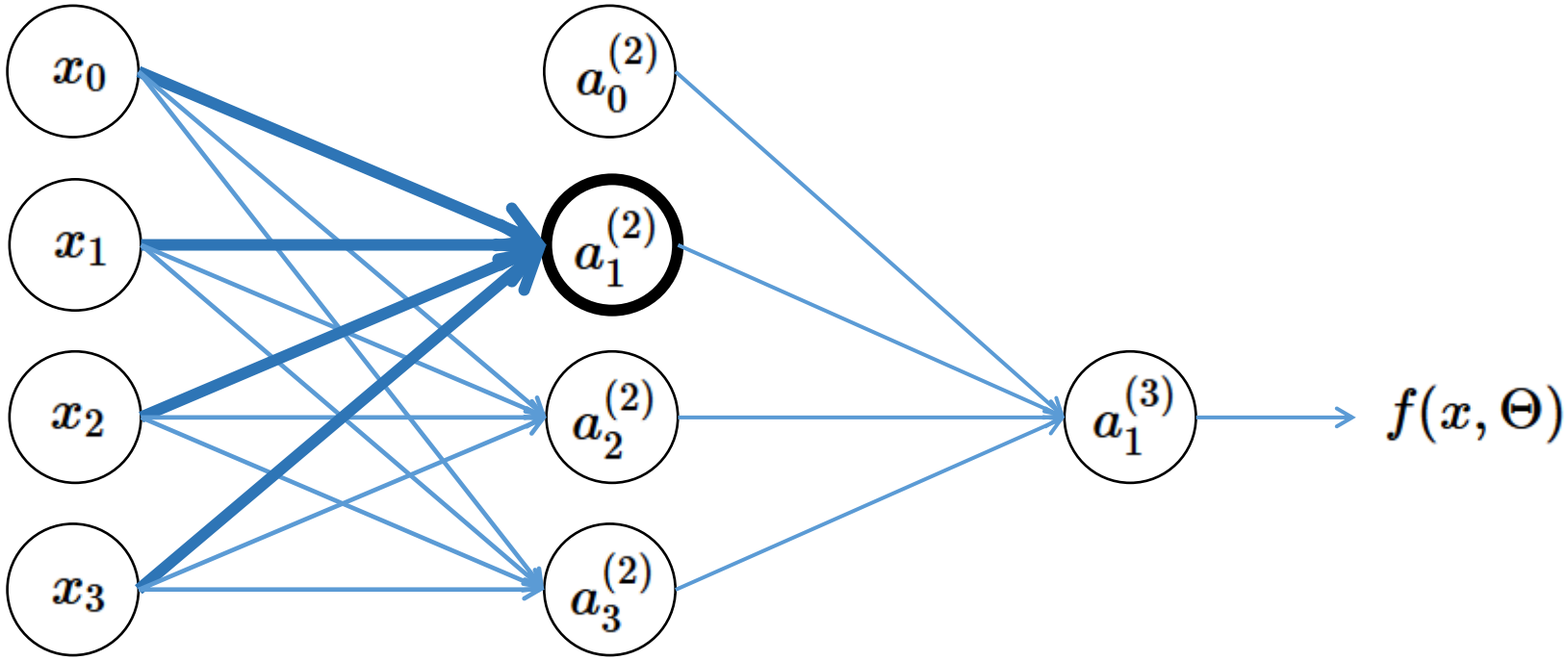


Model: $f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

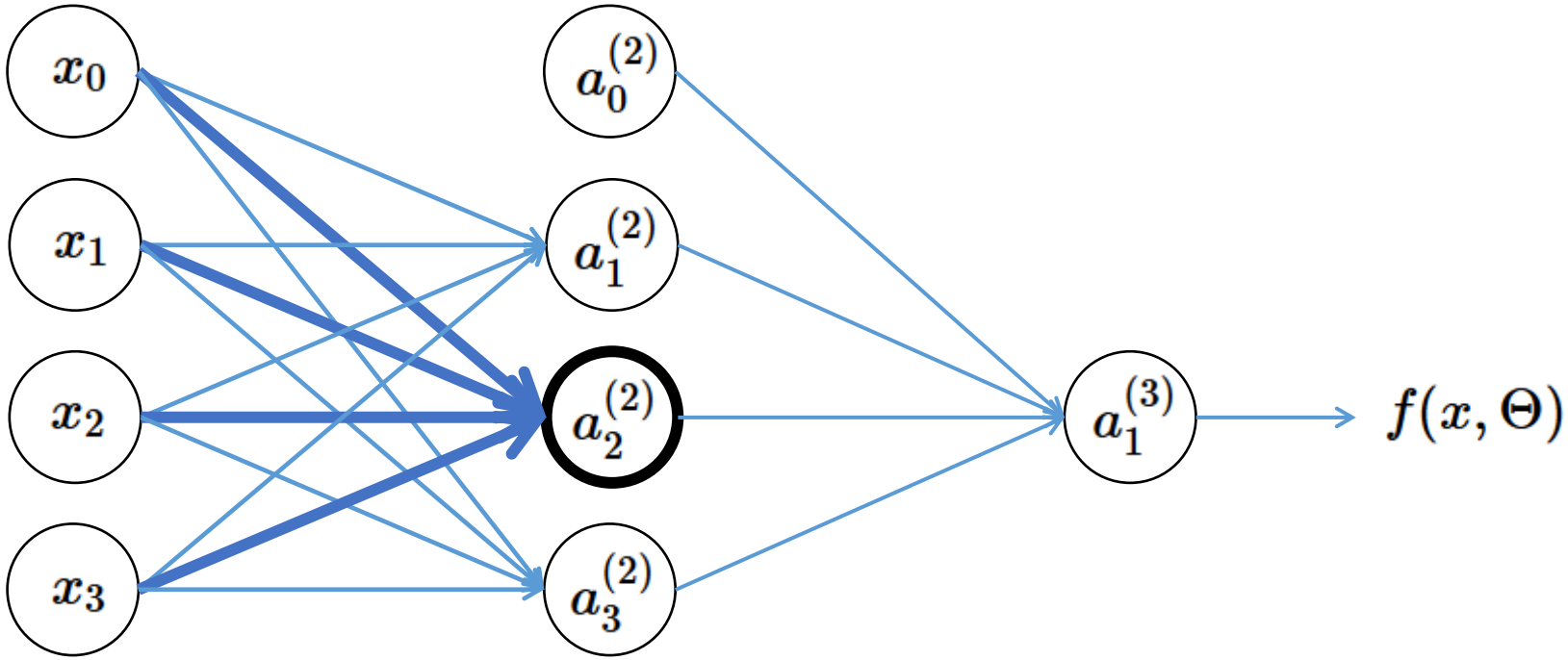


Model: $f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

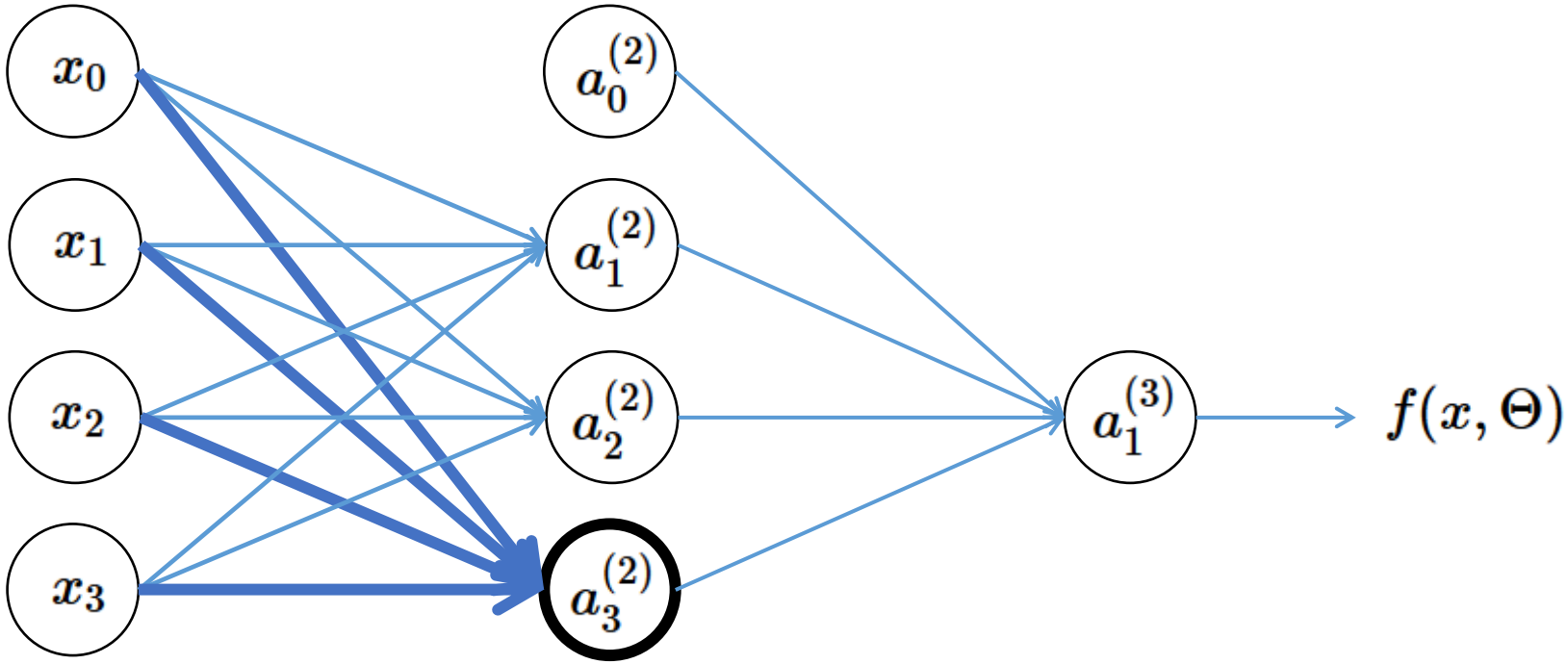


Model: $f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

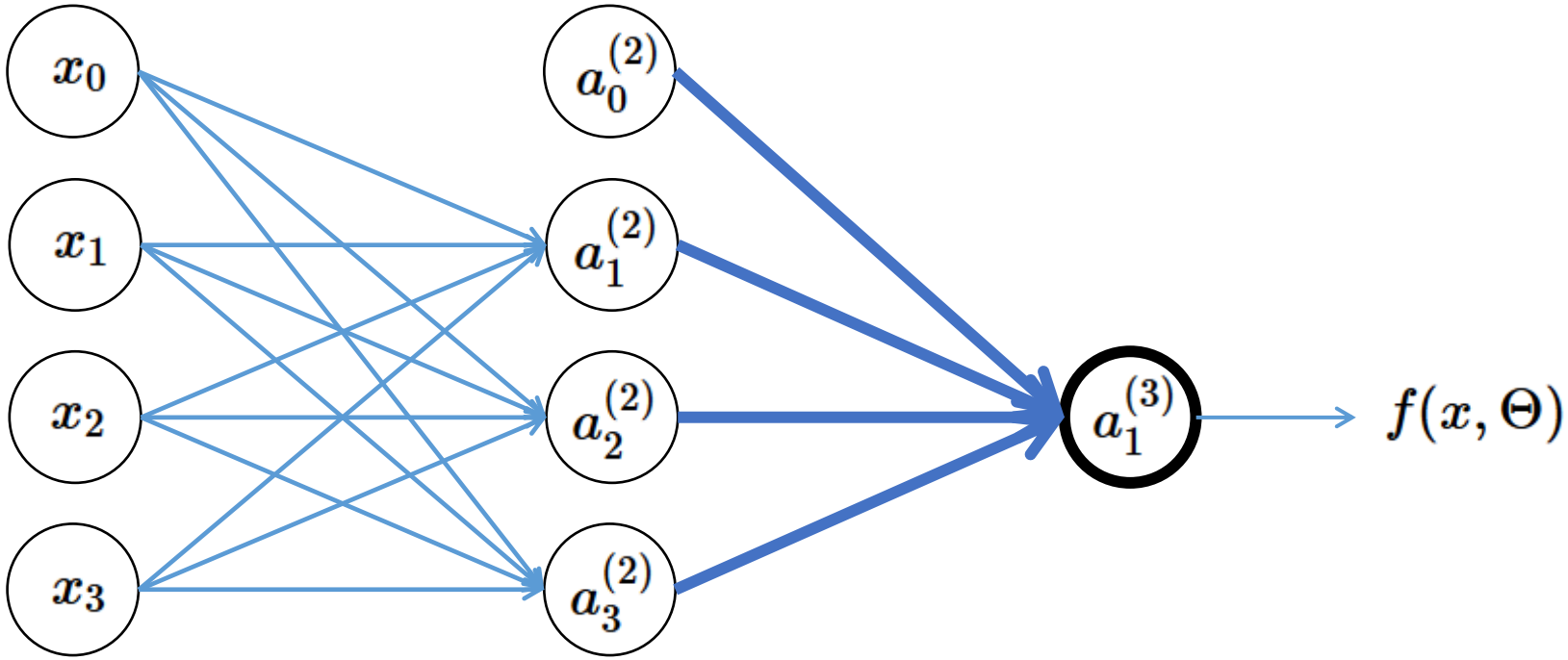


Model: $f(x,\Theta) = g(\Theta_{10}^{(2)} a_0 + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$



Cost function

Logistic regression:

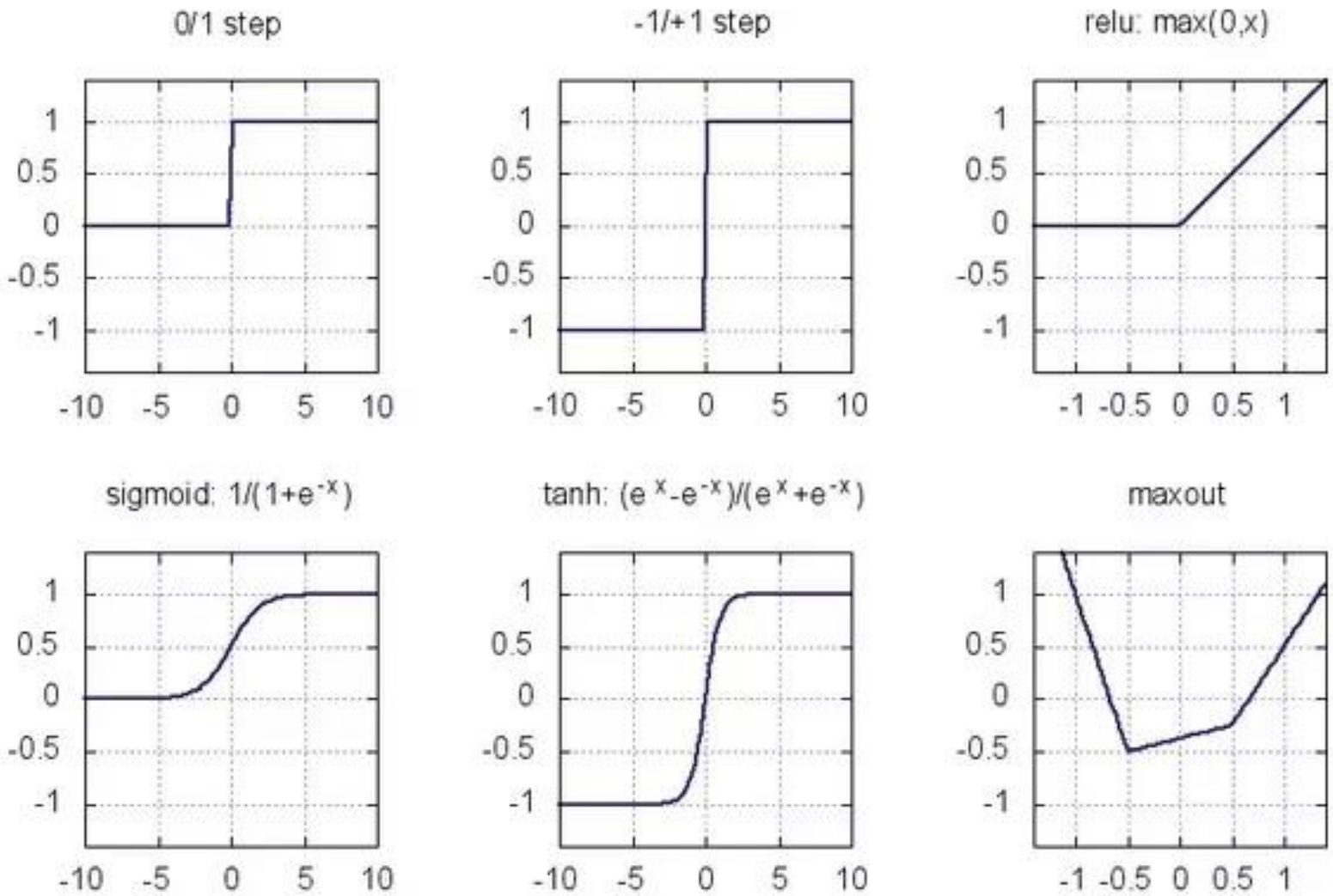
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Neural network:

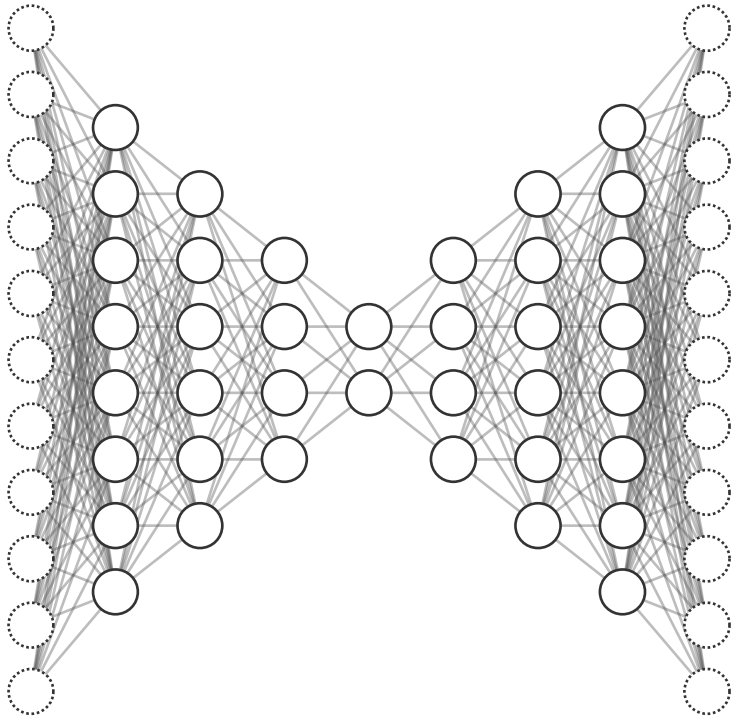
$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log((h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$

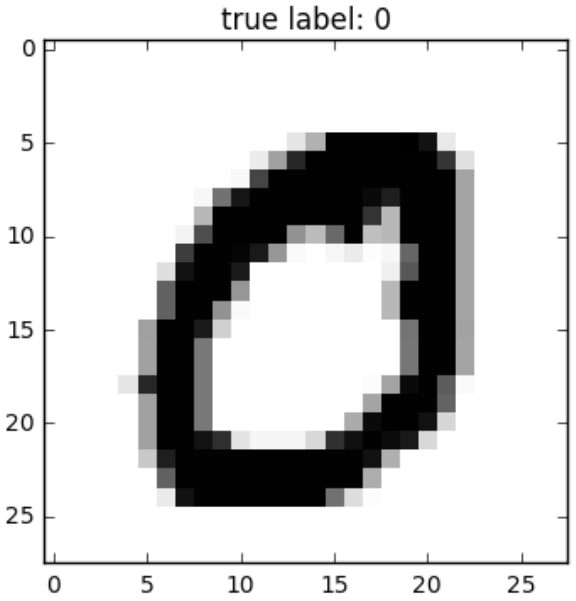
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

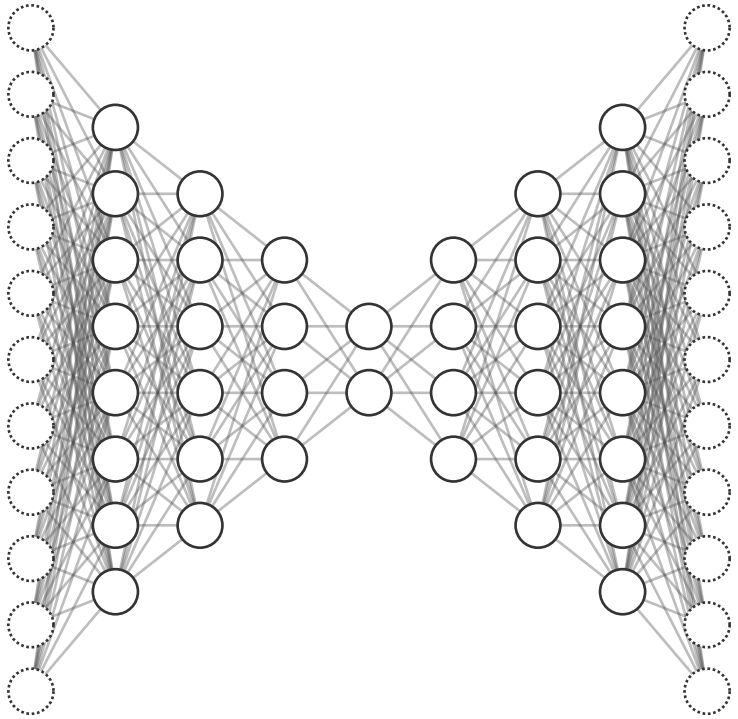


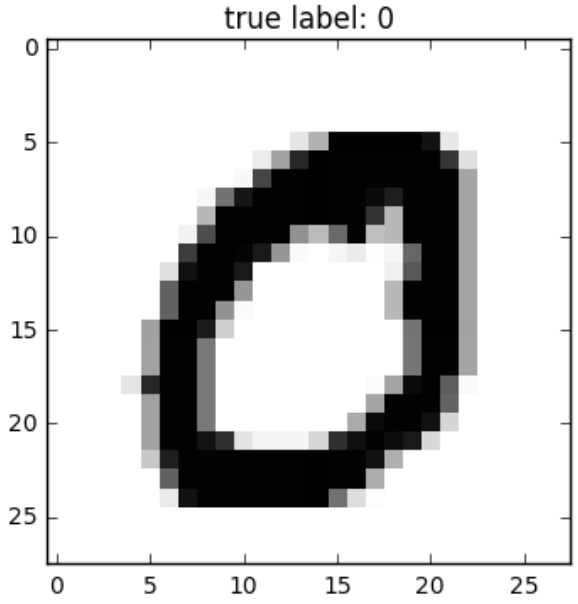
Activation functions



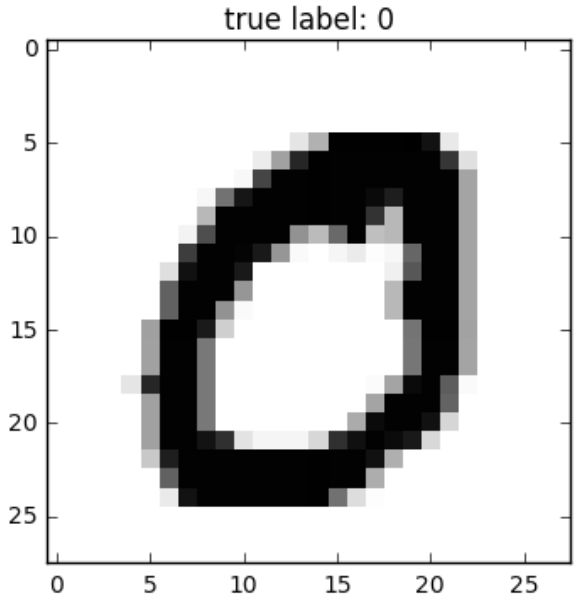
Auto-encoders

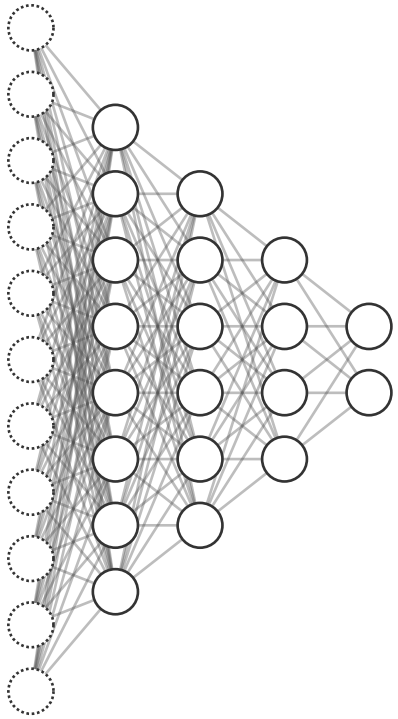


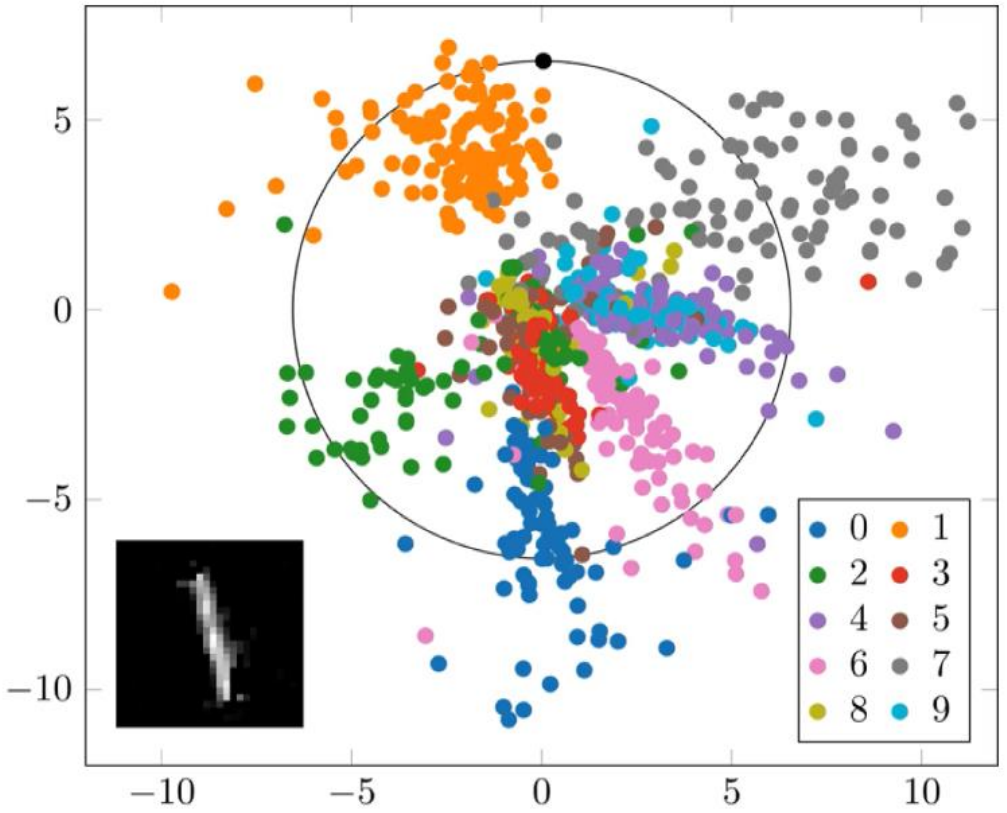




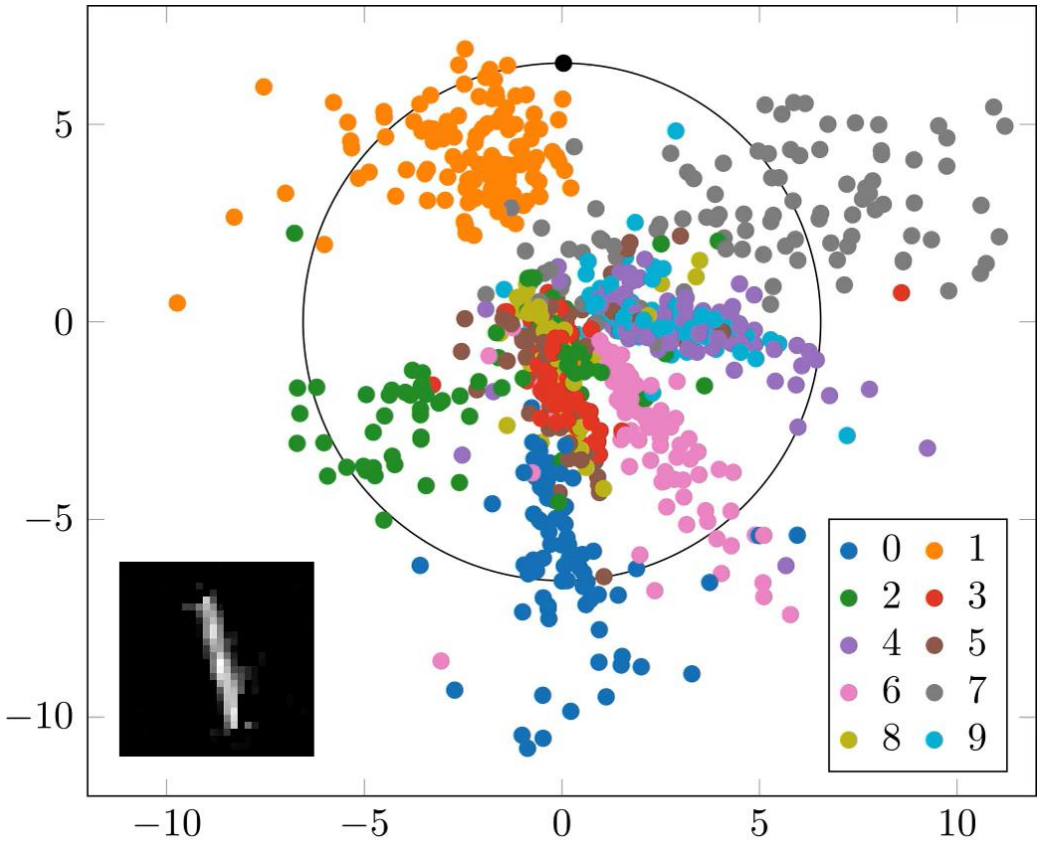
Auto-encoders

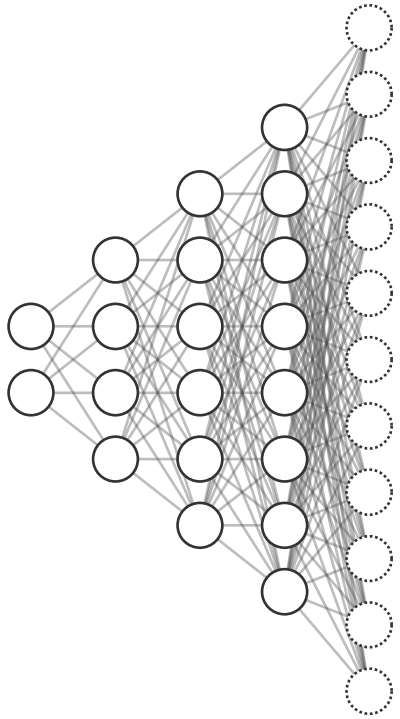


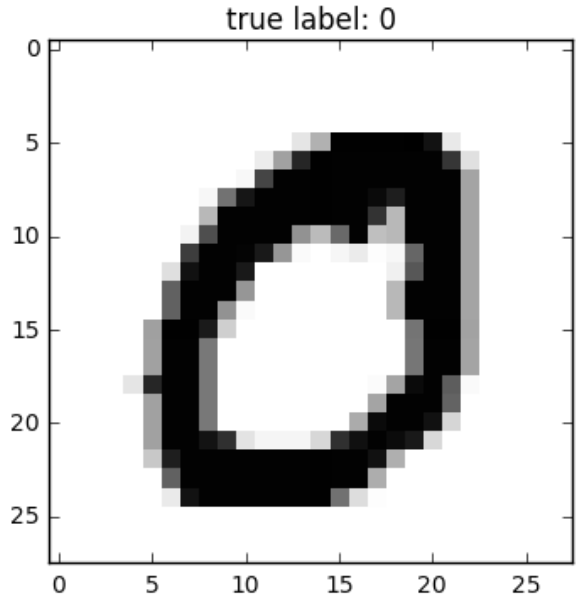


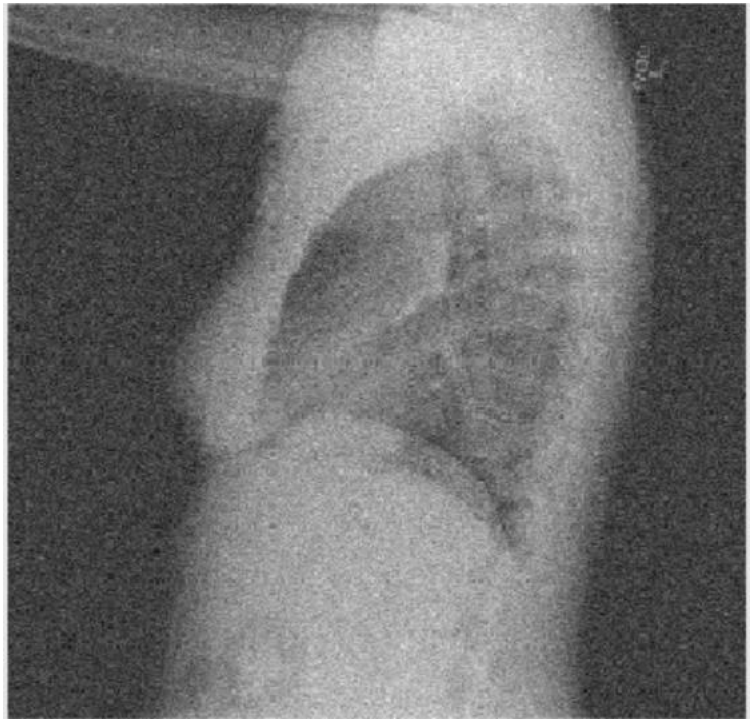


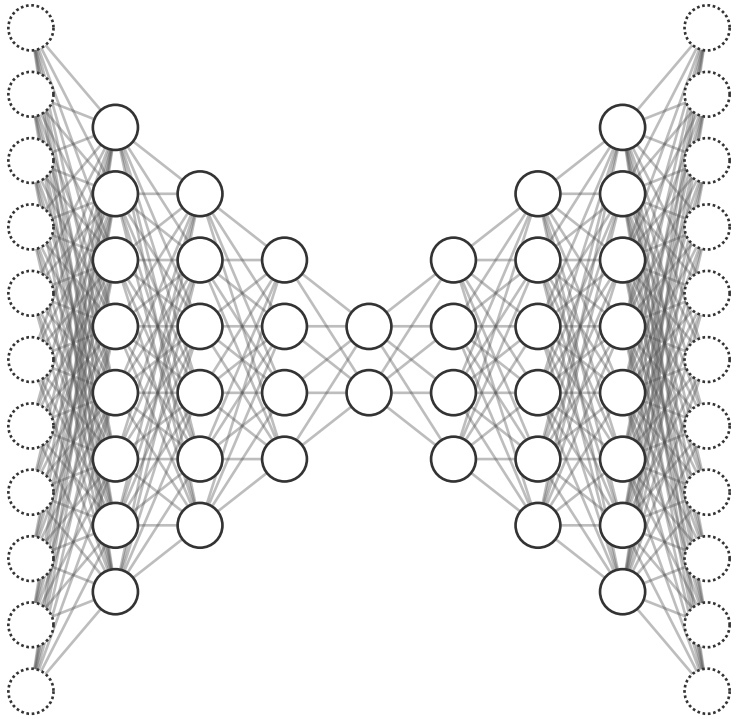
Auto-encoders

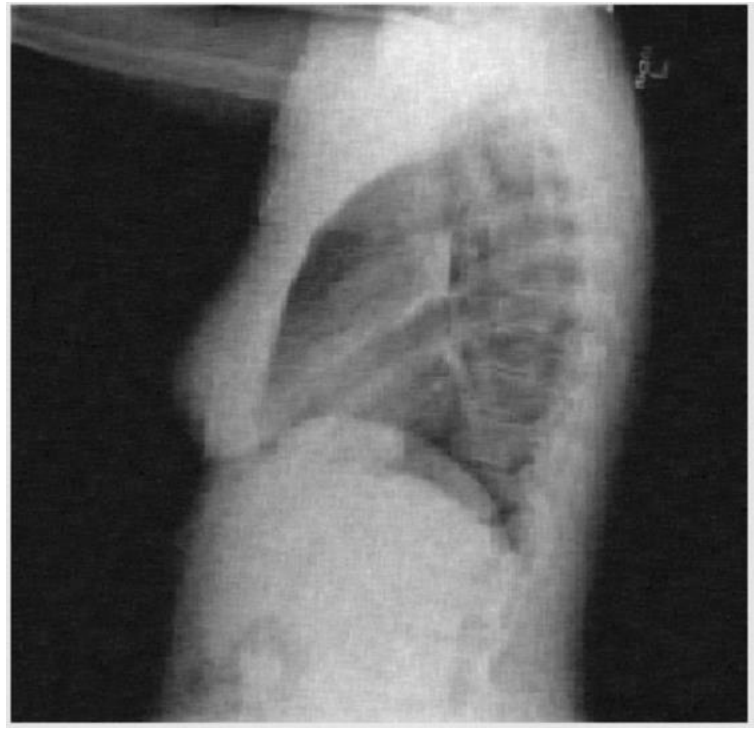

5
0
−5
−10
−10
−5
0
5
10
0 1
2 3
4 5
6 7
8 9




true label: 0
0
5
10
15
20
25
0
5
10
15
20
25



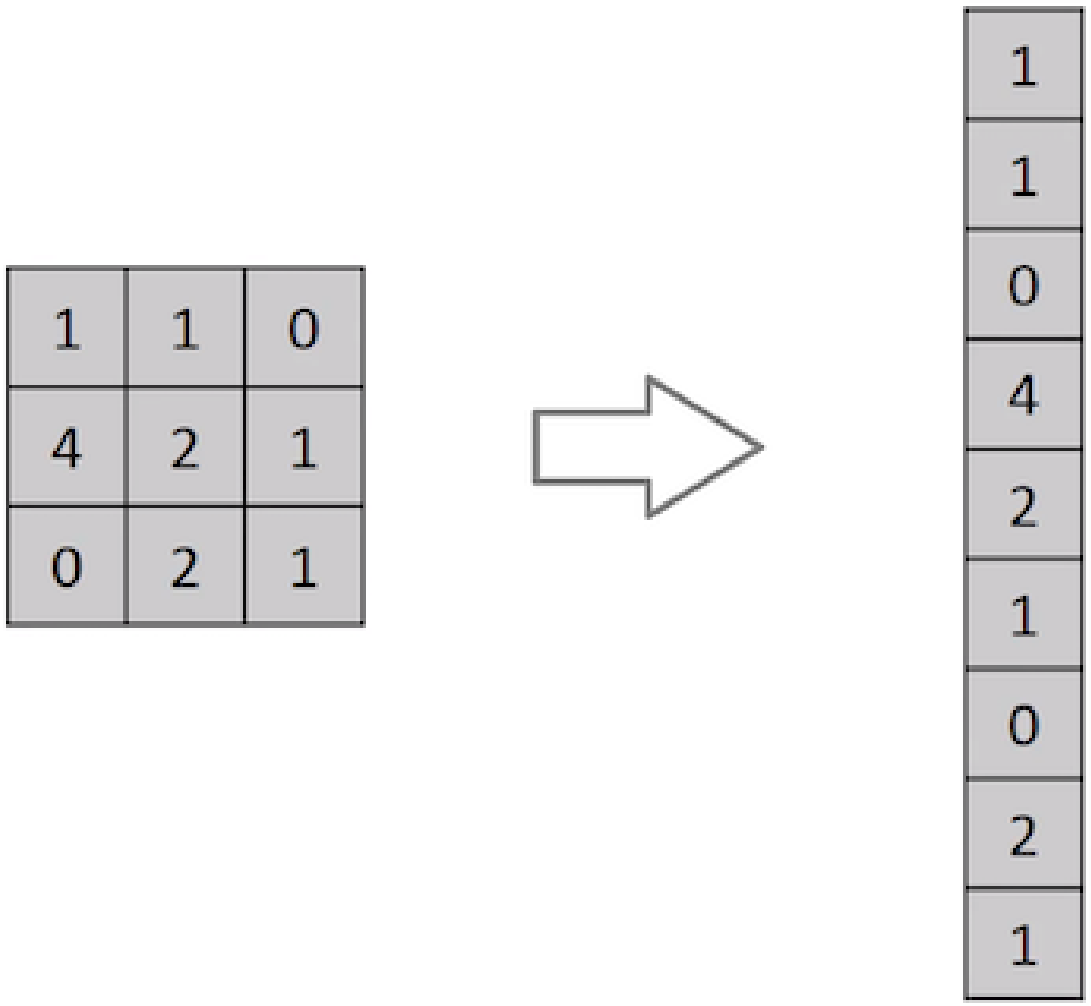


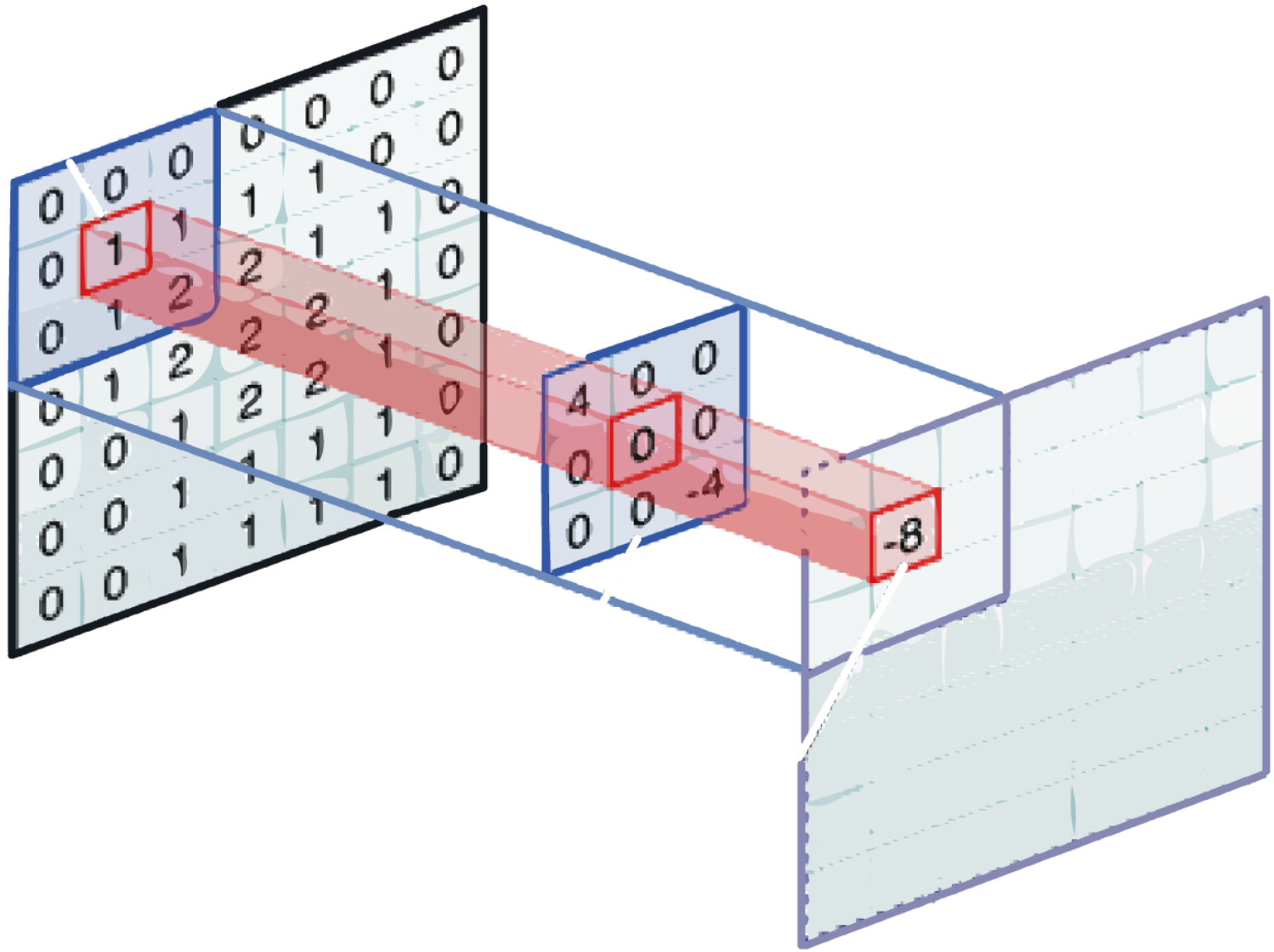


Denoising auto-encoders

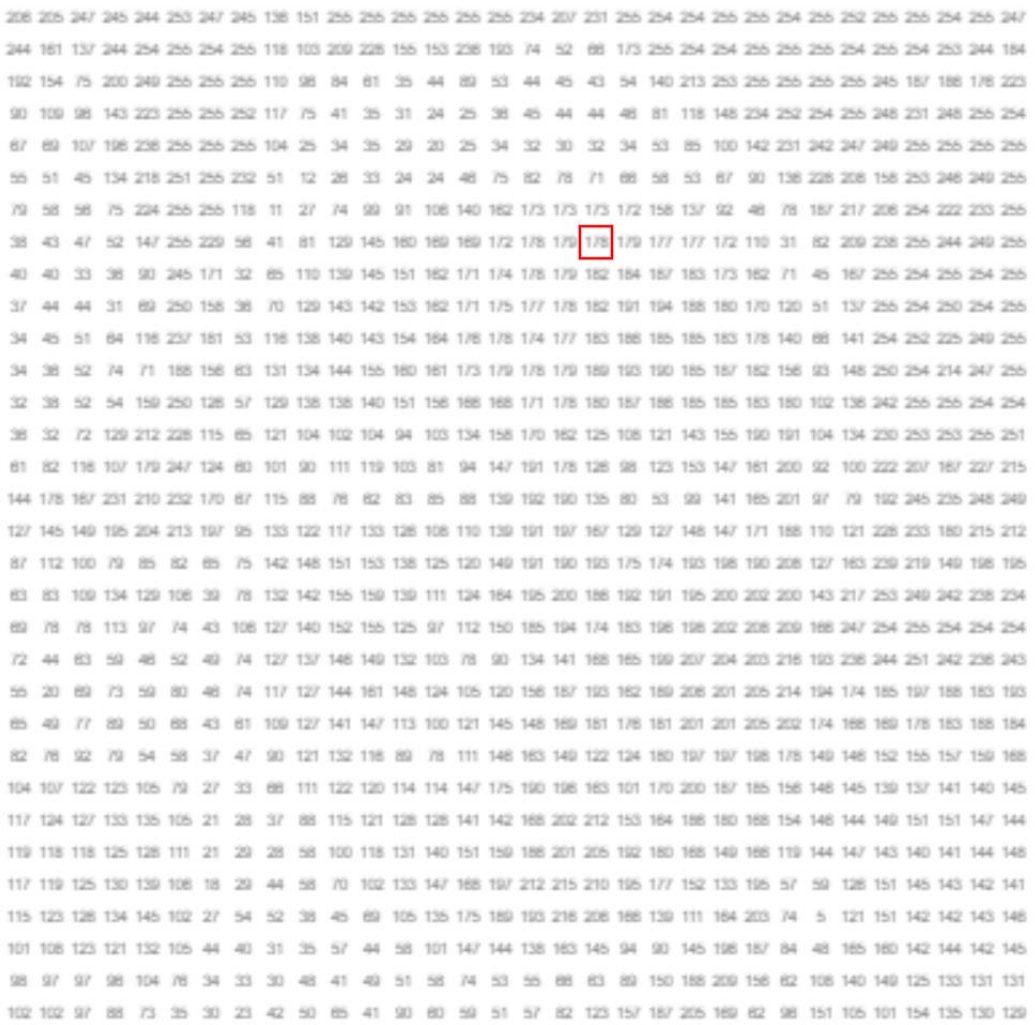
206 205 247 245 244 253 247 245 136 151 255 255 255 255 255 255 254 207 231 255 254 254 255 255 254 255 252 255 255 254 255 247
244 161 137 244 254 255 254 255 118 103 209 228 155 153 236 193 74 52 66 173 255 254 254 255 255 255 254 255 254 253 244 184
192 154 75 200 249 255 255 255 110 98 84 61 35 44 89 53 44 46 43 54 140 213 253 255 255 255 255 255 245 187 186 176 223
90 109 98 143 223 255 255 252 117 75 41 35 31 24 25 38 46 44 44 46 81 118 148 234 252 254 255 248 231 248 255 254
67 69 107 196 236 255 255 255 104 25 34 35 29 20 25 34 32 30 32 34 53 85 100 142 231 242 247 249 255 255 255 255
55 51 46 134 218 251 255 232 51 12 26 33 24 24 46 75 82 78 71 66 58 53 67 90 136 228 208 158 253 246 249 255
79 58 56 75 224 255 255 118 11 27 74 99 91 106 140 162 173 173 173 172 158 137 92 46 78 187 217 206 254 222 233 255
38 43 47 52 147 255 229 56 41 81 129 145 160 169 169 172 178 179 178 179 177 177 172 110 31 82 209 238 255 244 249 255
40 40 33 36 90 245 171 32 65 110 139 145 151 162 171 174 178 179 182 184 187 183 173 162 71 46 167 255 254 255 254 255
37 44 44 31 69 250 158 36 70 129 143 142 153 162 171 175 177 178 182 191 194 188 180 170 120 51 137 255 254 250 254 255
34 46 51 64 116 237 181 53 116 138 140 143 154 164 176 178 174 177 183 186 185 185 183 178 140 66 141 254 252 225 249 255
34 36 52 74 71 188 156 63 131 134 144 155 160 161 173 179 178 179 189 193 190 185 187 182 156 93 148 250 254 214 247 255
32 38 52 54 159 250 126 57 129 138 138 140 151 156 166 168 171 178 180 187 186 185 185 183 180 102 136 242 255 255 254 254
36 32 72 129 212 228 115 65 121 104 102 104 94 103 134 158 170 162 125 108 121 143 155 190 191 104 134 230 253 253 255 251
61 82 116 107 179 247 124 60 101 90 111 119 103 81 94 147 191 178 126 98 123 153 147 161 200 92 100 222 207 167 227 215
144 178 167 231 210 232 170 67 115 88 76 62 83 85 88 139 192 190 135 80 53 99 141 165 201 97 79 192 245 235 248 249
127 145 149 196 204 213 197 95 133 122 117 133 126 108 110 139 191 197 167 129 127 148 147 171 188 110 121 228 233 180 215 212
87 112 100 79 85 82 65 75 142 148 151 153 138 125 120 149 191 190 193 175 174 193 198 190 208 127 163 239 219 149 198 196
63 83 109 134 129 106 39 78 132 142 155 159 139 111 124 164 196 200 186 192 191 196 200 202 200 143 217 253 249 242 238 234
69 78 78 113 97 74 43 106 127 140 152 155 125 97 112 150 185 194 174 183 196 198 202 206 209 166 247 254 255 254 254 254
72 44 63 59 46 52 49 74 127 137 146 149 132 103 78 90 134 141 168 185 199 207 204 203 216 193 236 244 251 242 236 243
55 20 69 73 59 80 46 74 117 127 144 161 148 124 105 120 156 187 193 162 189 206 201 205 214 194 174 185 197 188 183 193
65 49 77 89 50 68 43 61 109 127 141 147 113 100 121 145 148 169 181 176 181 201 201 205 202 174 166 169 178 183 188 184
82 76 92 79 54 58 37 47 90 121 132 116 89 78 111 146 163 149 122 124 180 197 197 198 178 149 146 152 155 157 159 168
104 107 122 125 105 79 27 33 68 111 122 120 114 114 147 175 190 196 163 101 170 200 187 185 156 146 145 139 137 141 140 145
117 124 127 133 135 105 21 28 37 88 115 121 128 128 141 142 168 202 212 153 164 186 180 168 154 146 144 149 151 151 147 144
119 118 118 125 128 111 21 29 28 58 100 118 131 140 151 159 186 201 205 192 180 168 149 166 119 144 147 143 140 141 144 148
117 119 125 130 139 106 18 29 44 58 70 102 133 147 168 197 212 215 210 196 177 152 133 196 57 59 126 151 145 143 142 141
115 123 126 134 145 102 27 54 52 38 46 69 105 135 175 189 193 216 206 166 139 111 164 203 74 5 121 151 142 142 143 146
101 108 123 121 132 105 44 40 31 35 57 44 58 101 147 144 138 163 145 94 90 145 196 187 84 48 165 160 142 144 142 145
98 97 97 98 104 76 34 33 30 48 41 49 51 58 74 53 55 66 63 89 150 188 209 156 62 108 140 149 125 133 131 131
102 102 97 88 73 35 30 23 42 50 65 41 90 60 59 51 57 82 123 157 187 205 169 62 98 151 105 101 154 135 130 129

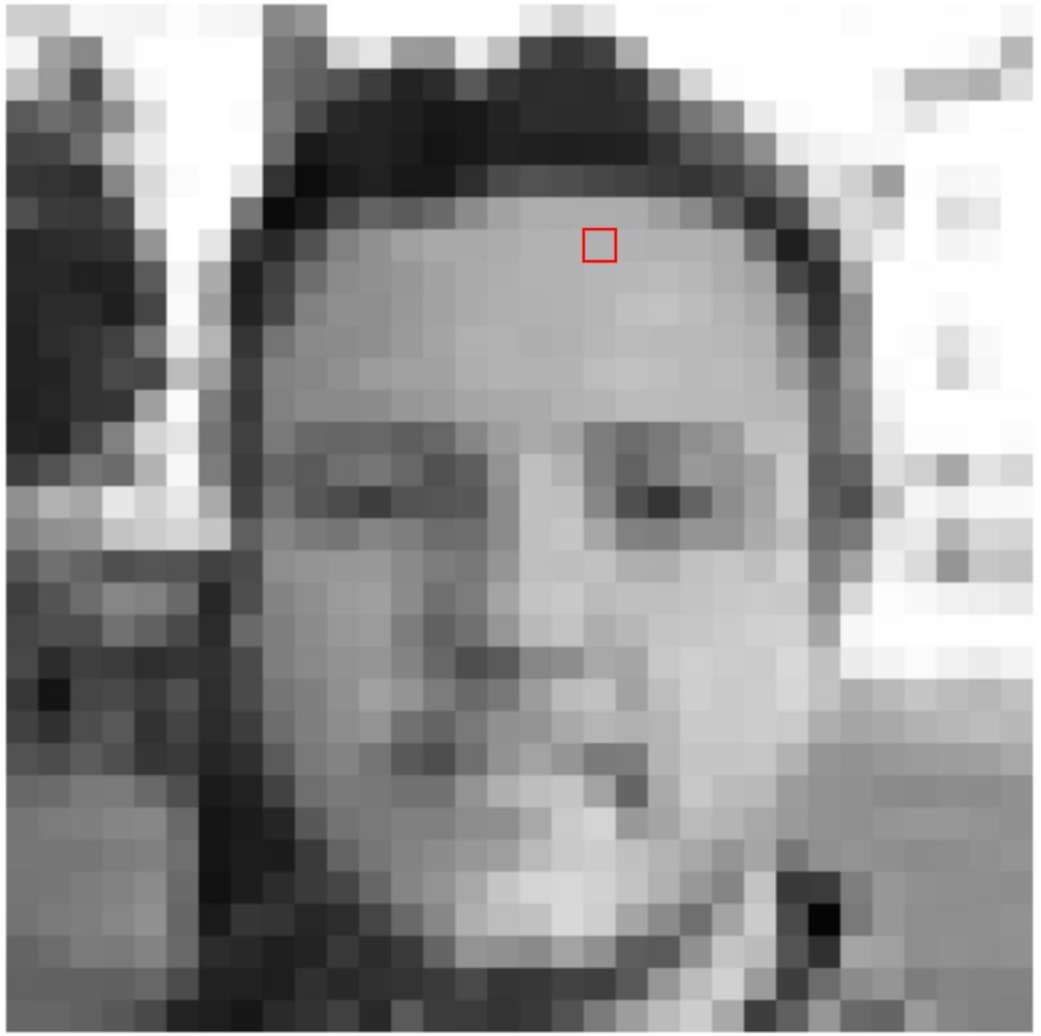
Convolutional filters: https://setosa.io/ev/image-kernels/


1 1 0
4 2 1
0 2 1
1
1
0
4
2
1
0
2
1

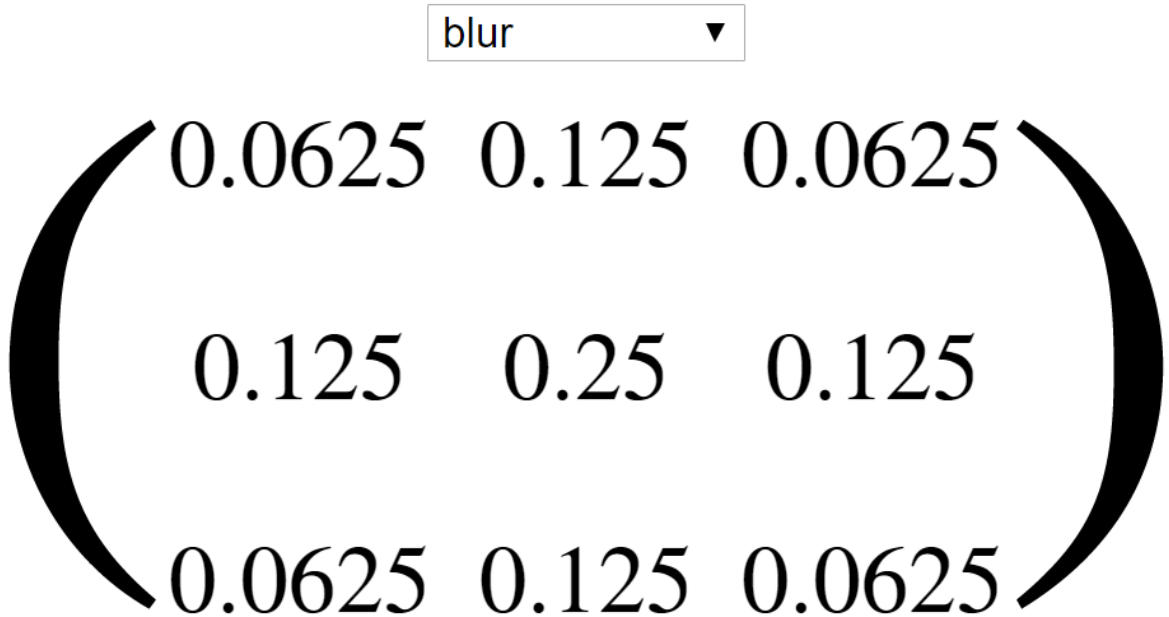


Convolutional filters: https://setosa.io/ev/image-kernels/

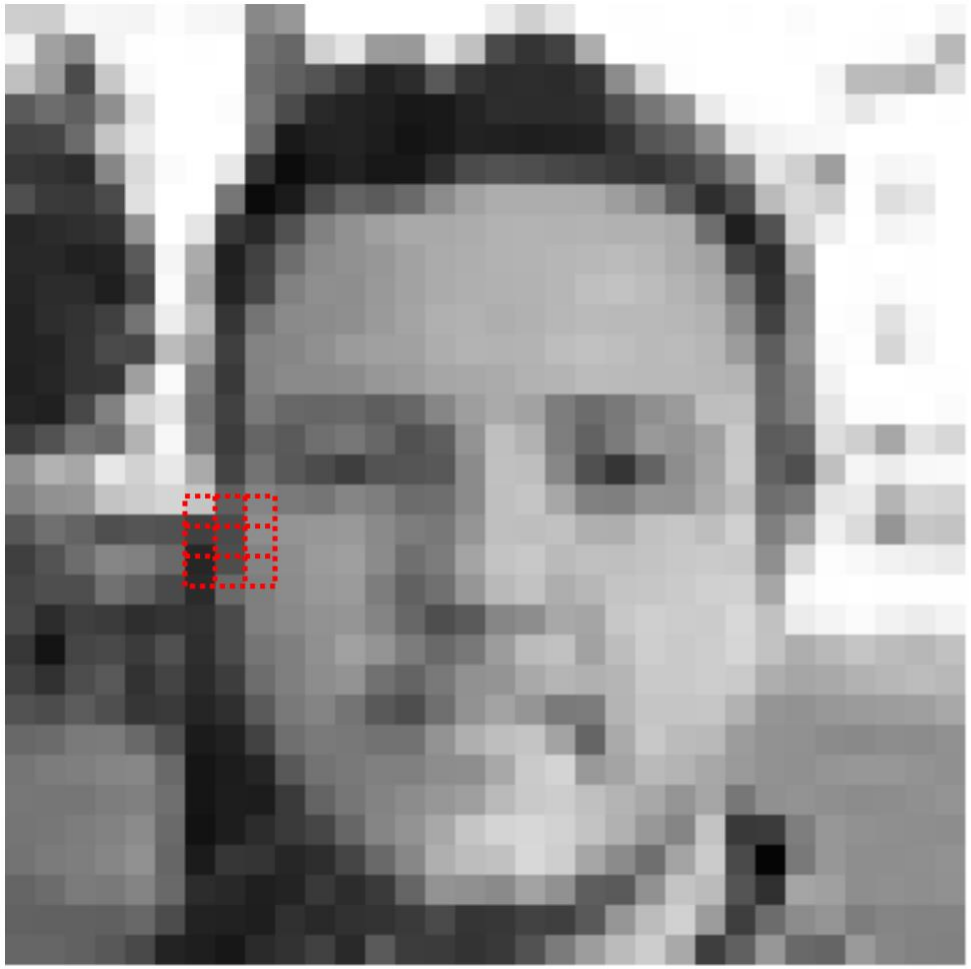




Convolutional filters: https://setosa.io/ev/image-kernels/



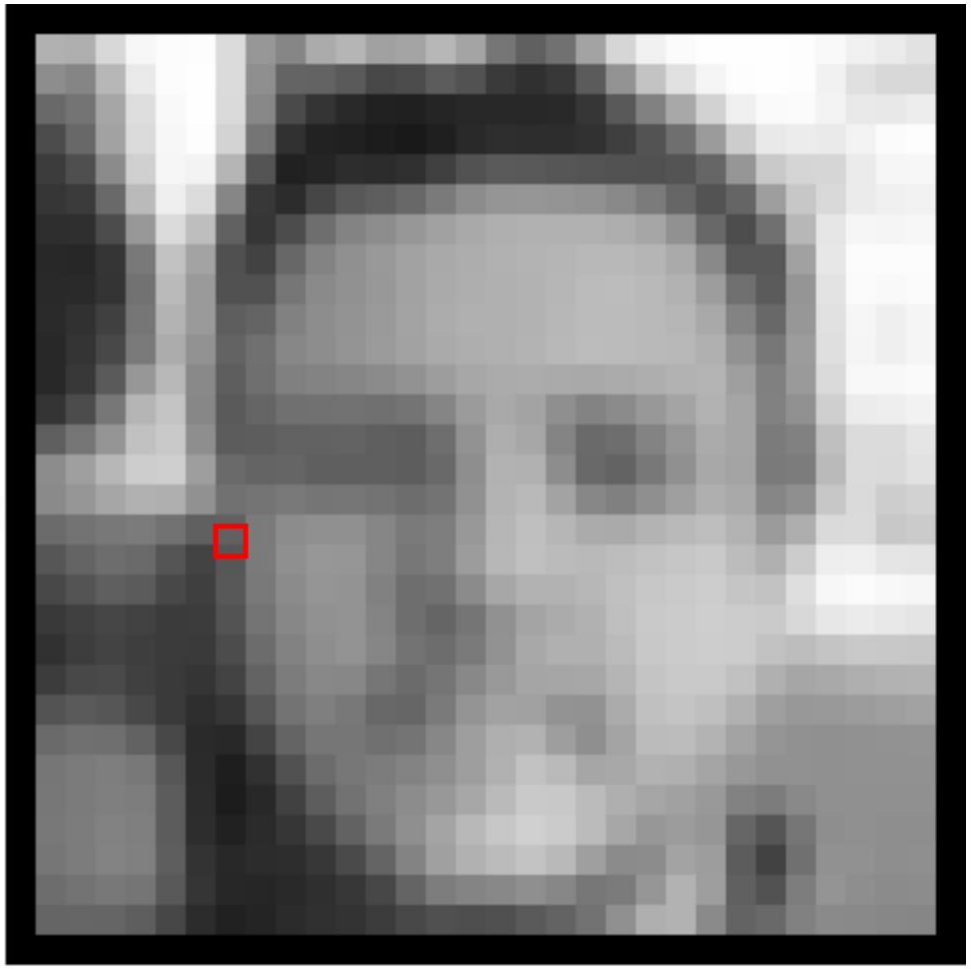
Convolutional filters: https://setosa.io/ev/image-kernels/



$$\Big(57 \times 0.0625 + 129 \times 0.125 + 138 \times 0.0625 + 65 \times 0.125 + 121 \times 0.25 + 104 \times 0.125 + 60 \times 0.0625 + 101 \times 0.125 + 90 \times 0.0625 \Big) = 102$$

kernel:

blur ▼



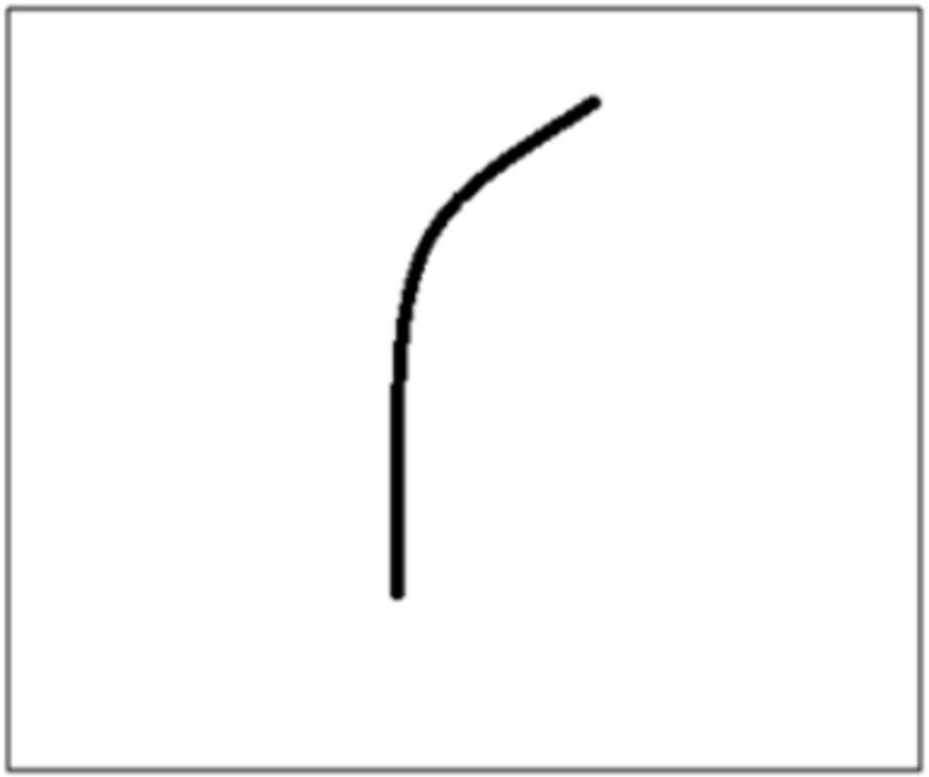
input image

output image

Convolutional filters: https://setosa.io/ev/image-kernels/

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

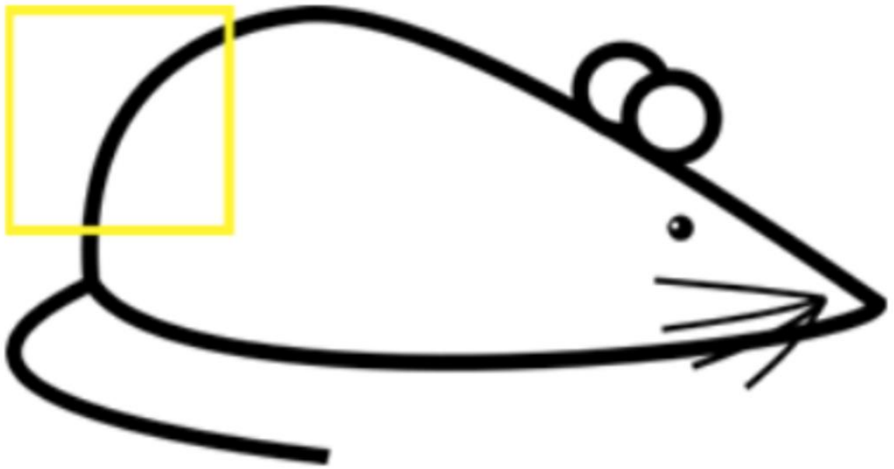
Pixel representation of filter



Visualization of a curve detector filter

Convolutional filters: https://setosa.io/ev/image-kernels/

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Convolutional filters: https://setosa.io/ev/image-kernels/

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

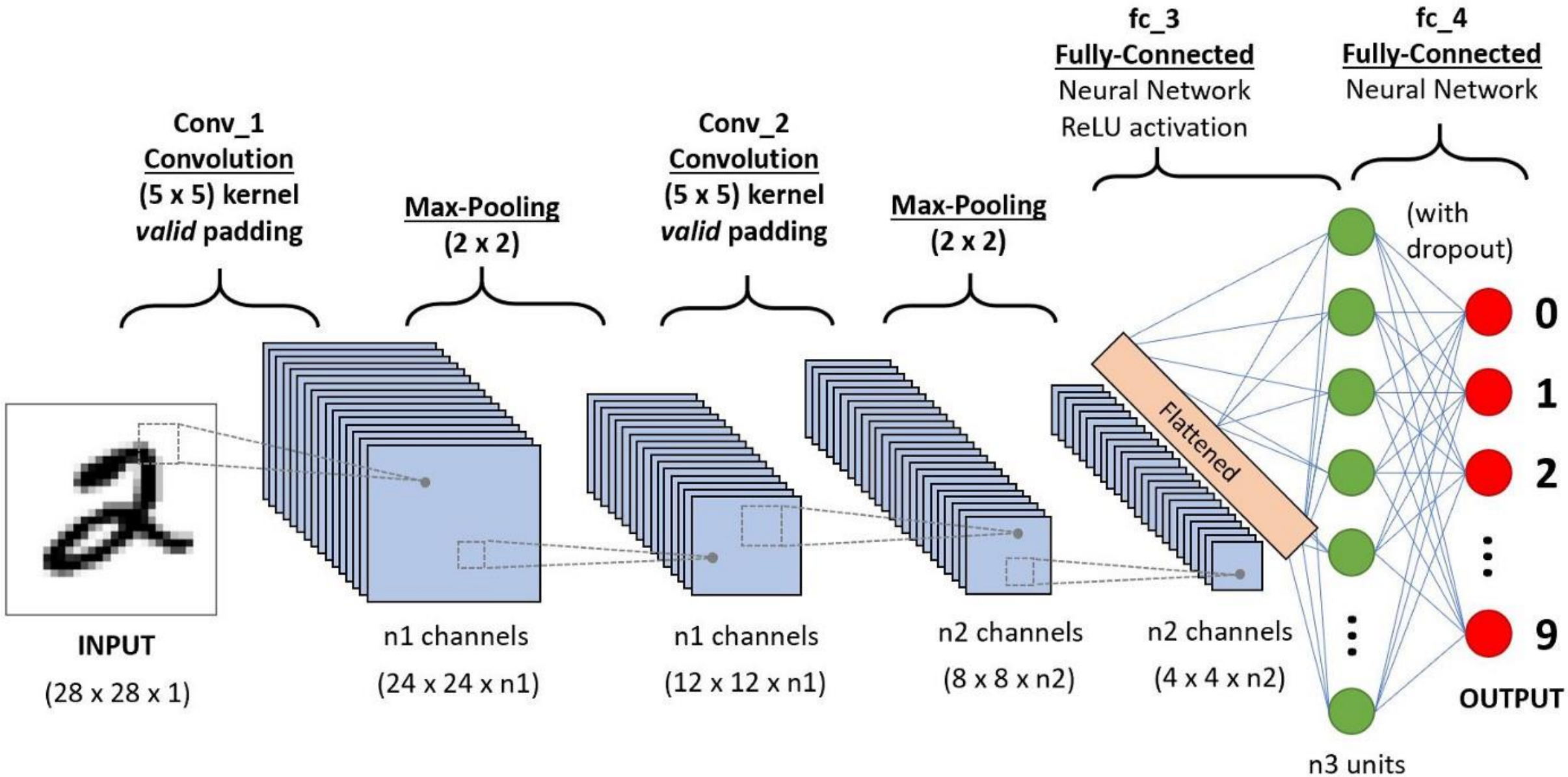
Pixel representation of receptive field

*

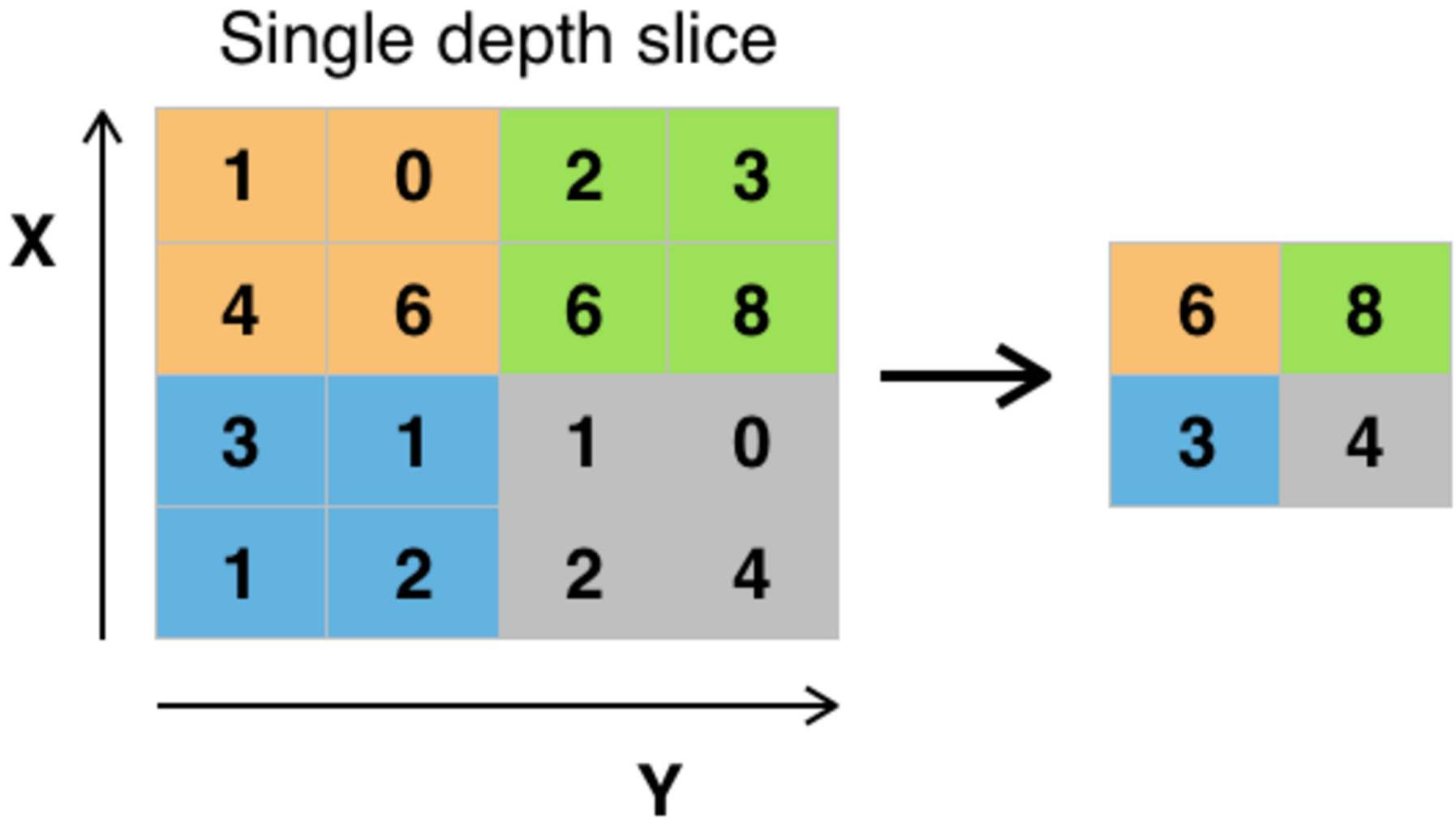
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Convolutional filters: https://setosa.io/ev/image-kernels/

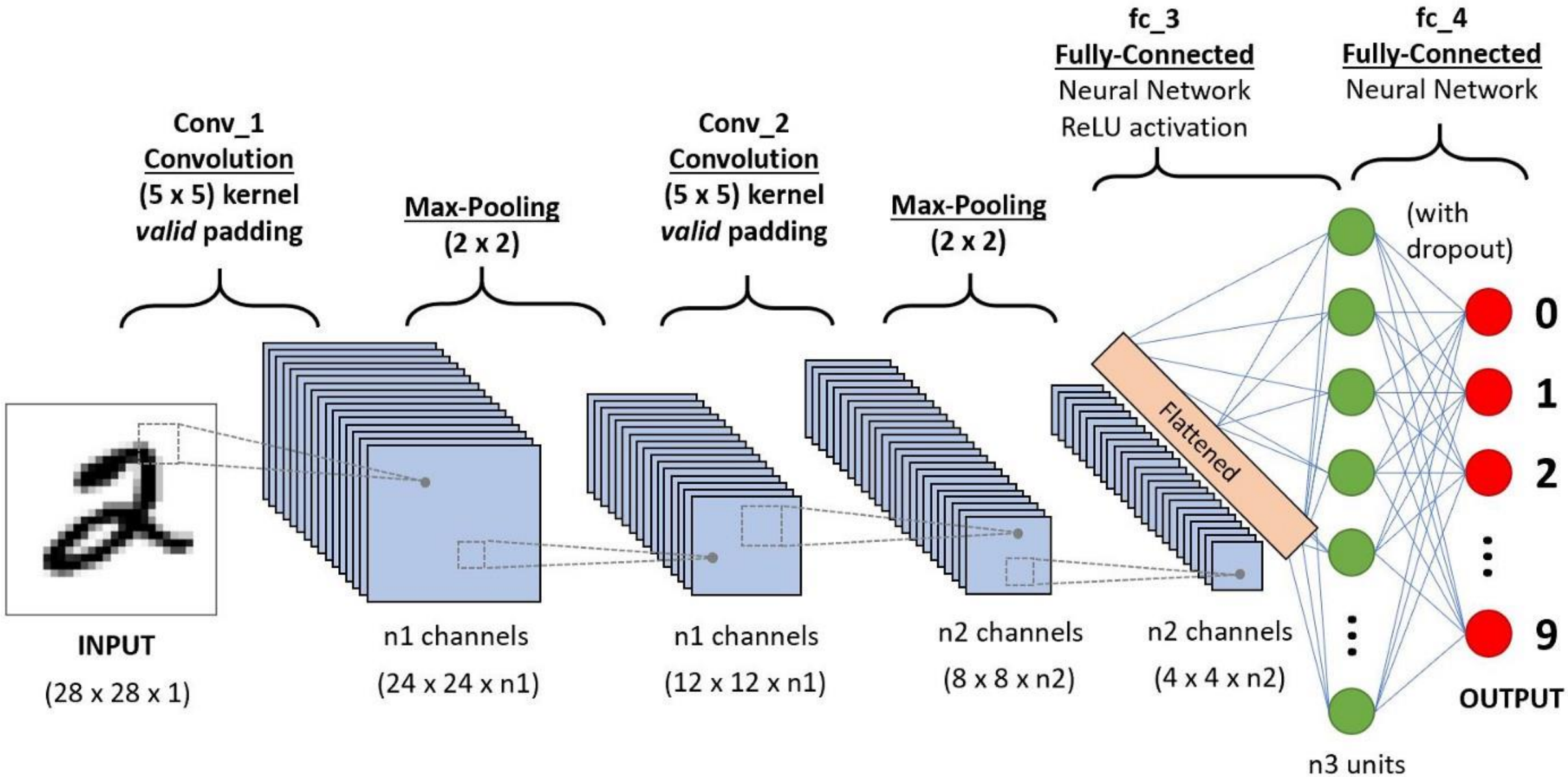


Convolutional neural network (CNN)

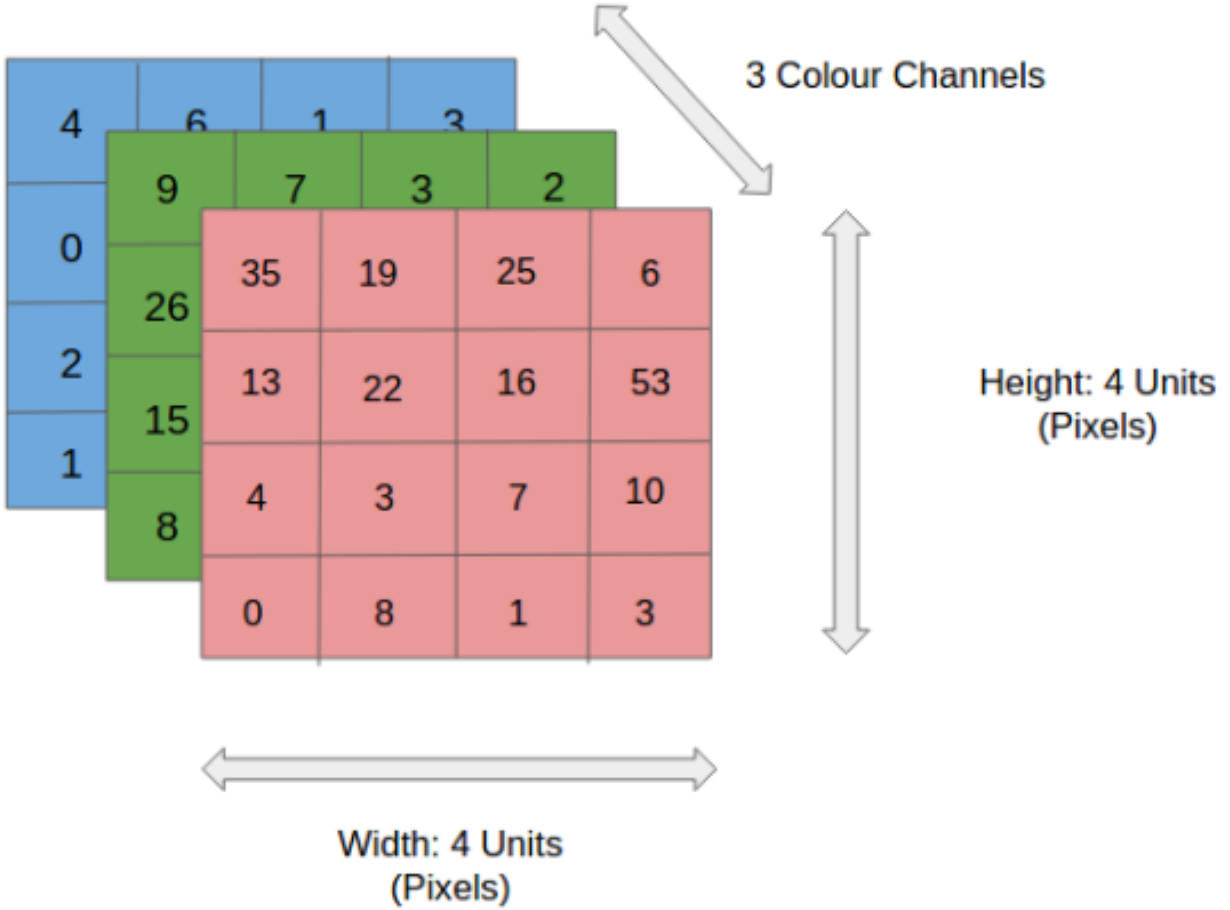


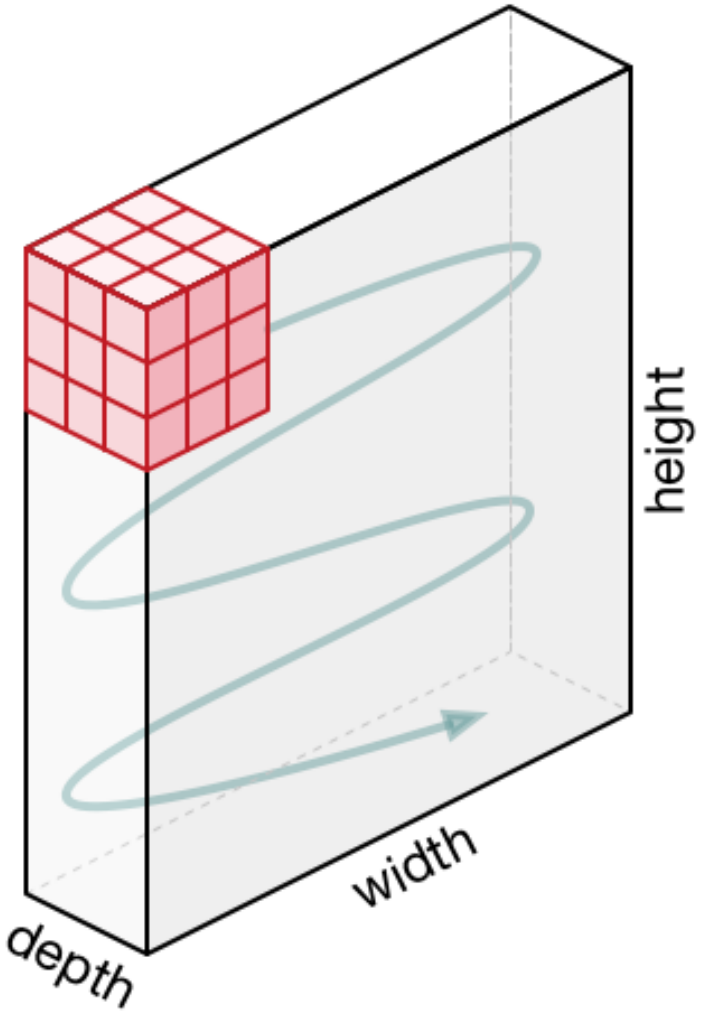
Example of Maxpool with a 2x2 filter and a stride of 2

Convolutional filters: https://setosa.io/ev/image-kernels/

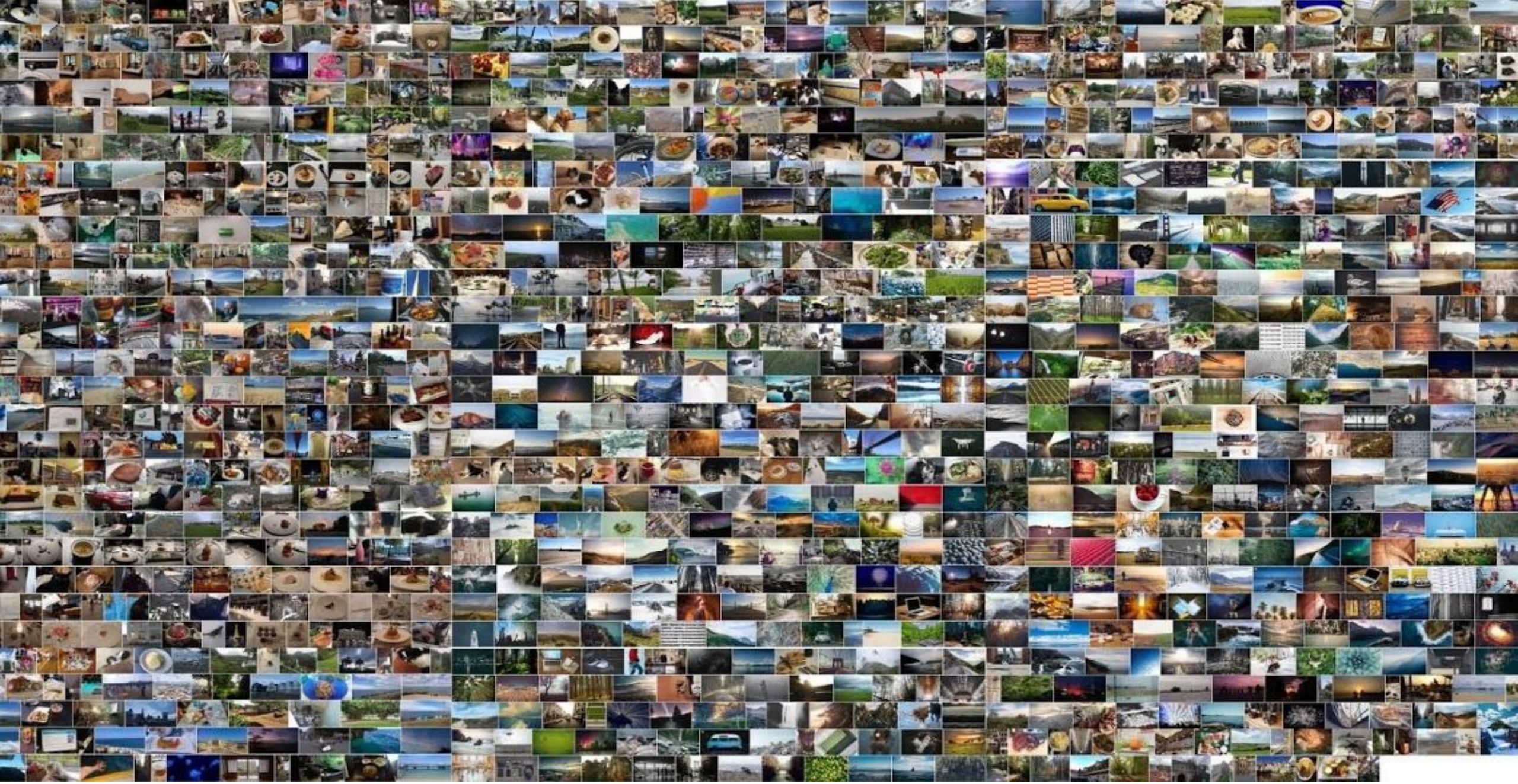


Convolutional neural network (CNN)

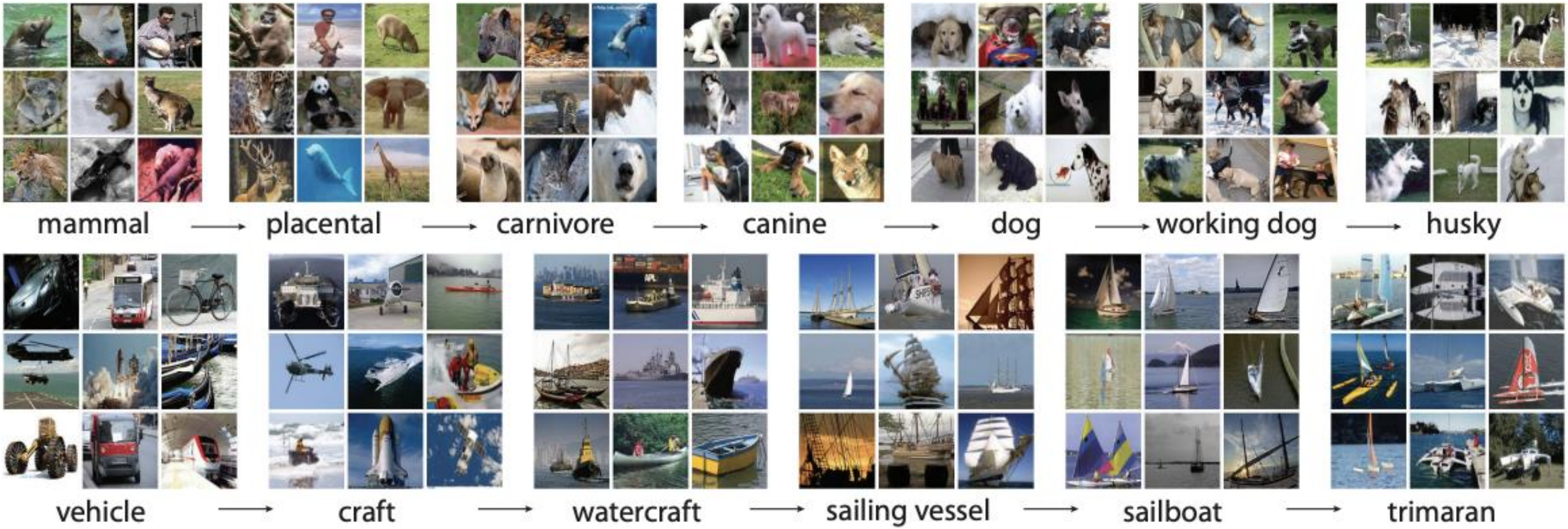




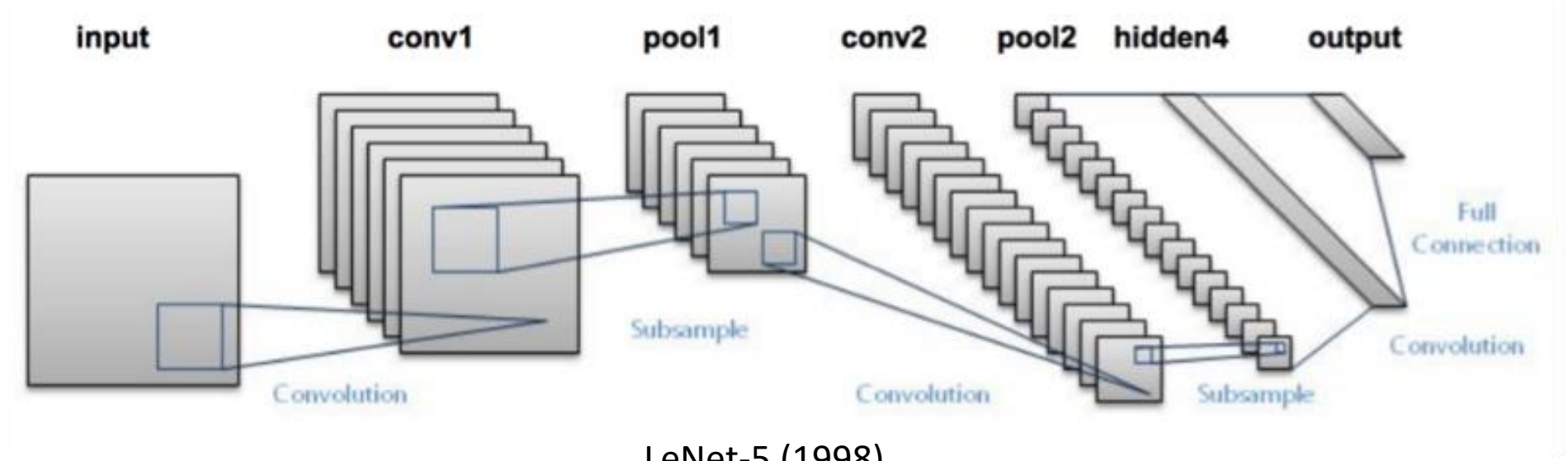
Convolutional filters: https://setosa.io/ev/image-kernels/



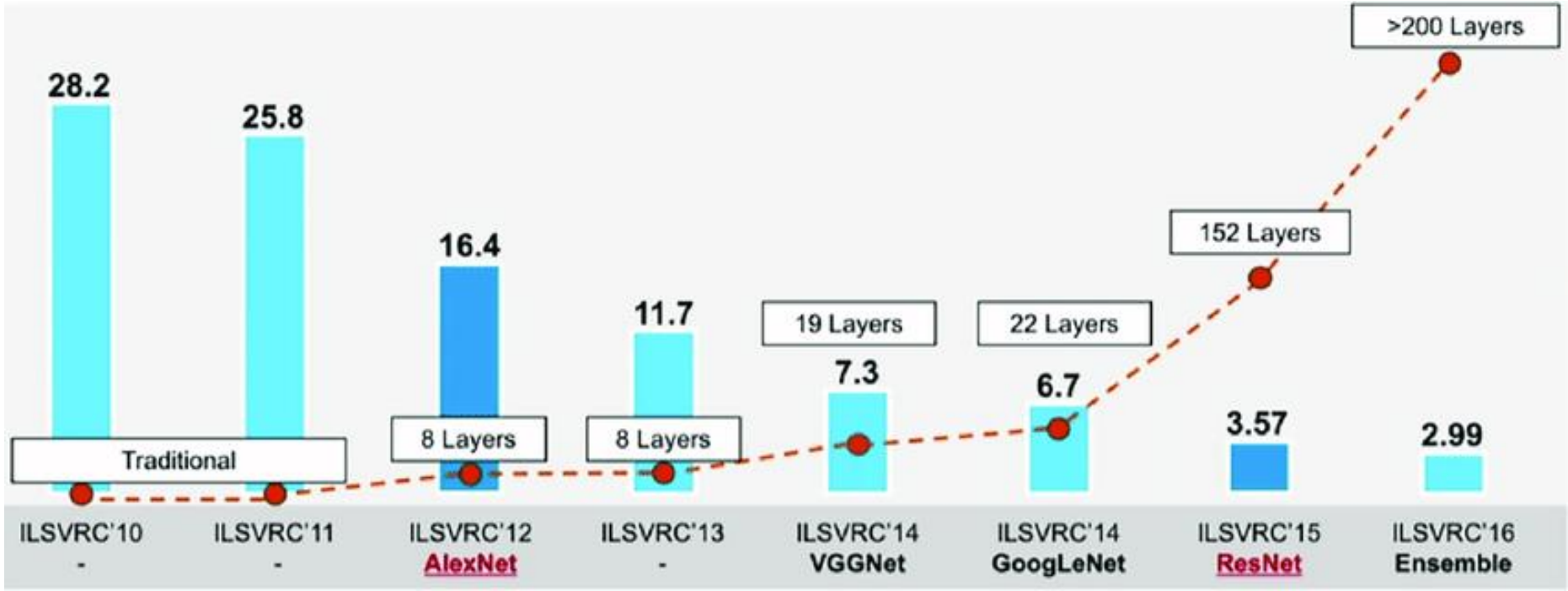
J. Deng *et al.*, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*



J. Deng *et al.*, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*



LeNet-5 (1998)

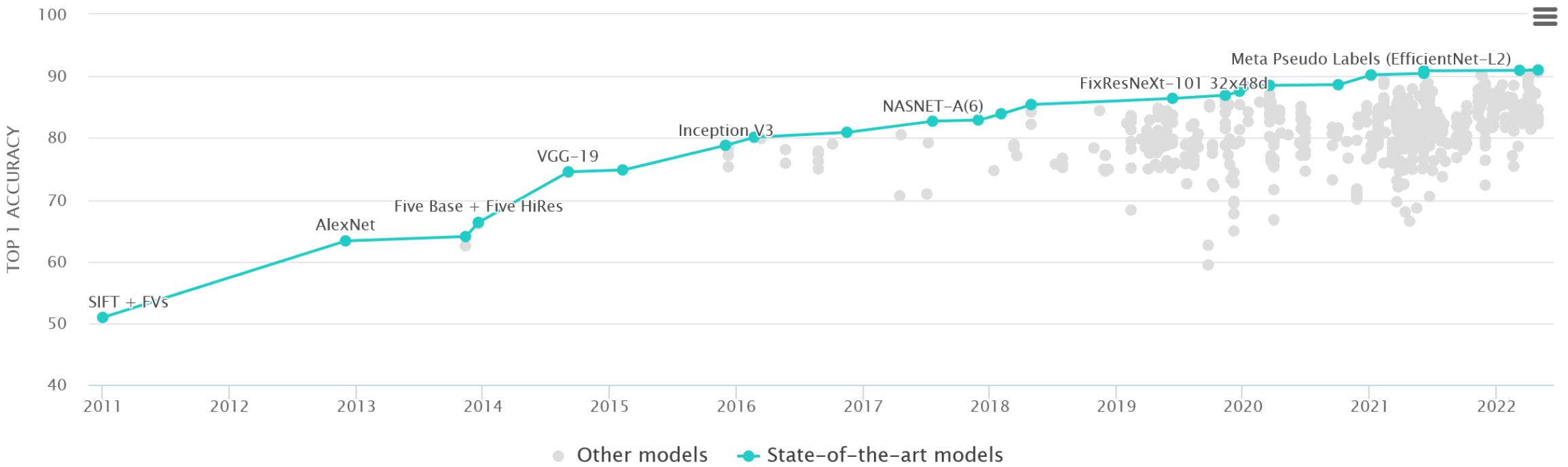


ImageNet Large Scale Visual Recognition Challenge: https://www.image-net.org/challenges/LSVRC/

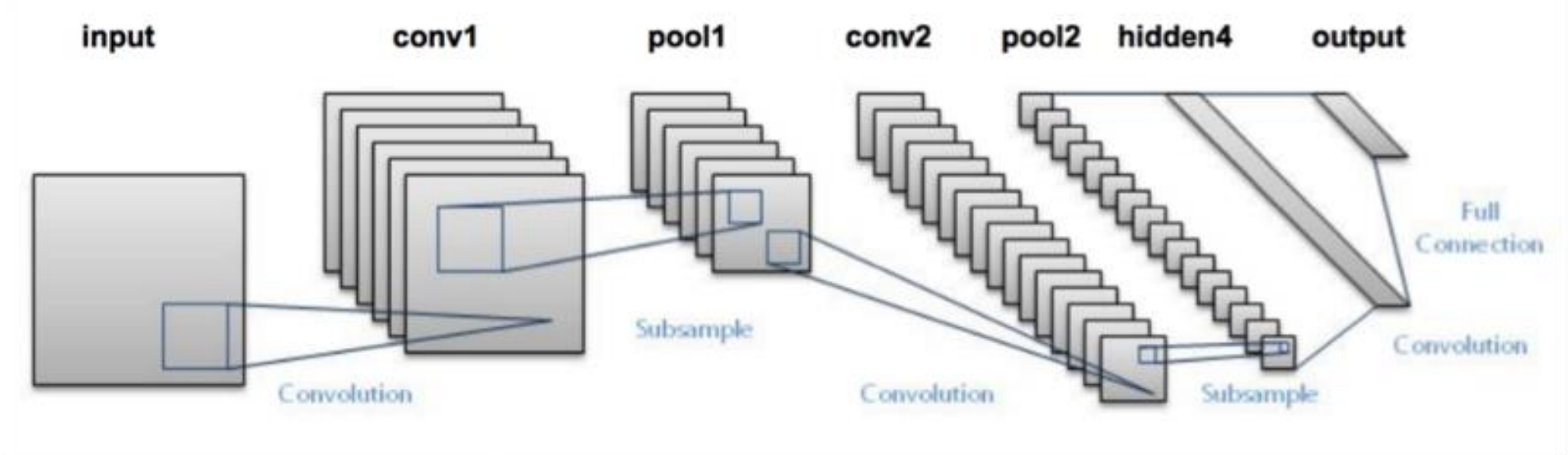
Image Classification on ImageNet

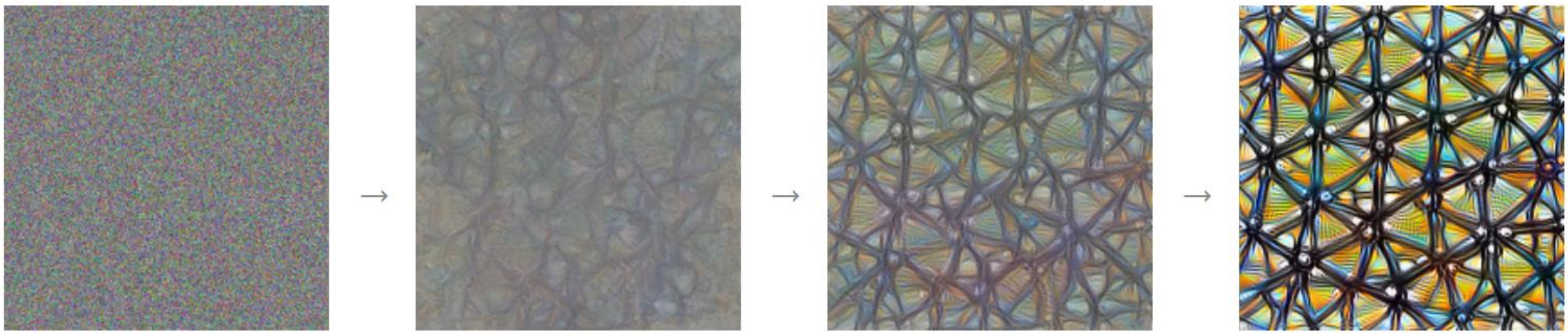
Leaderboard Dataset

View Top 1 Accuracy by Date for All models



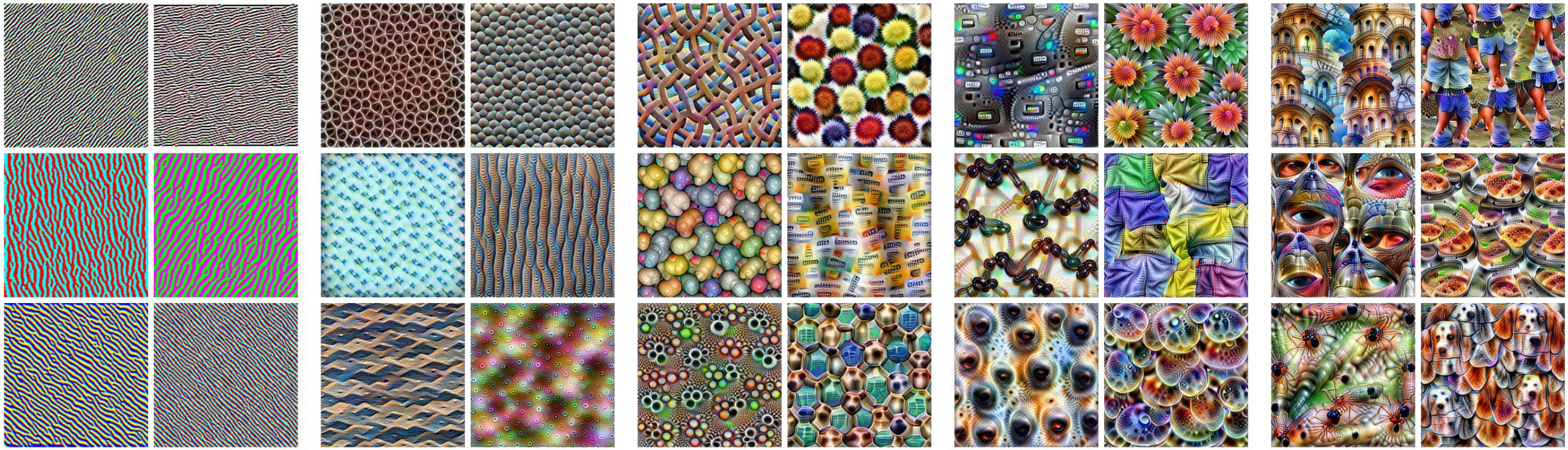
https://paperswithcode.com/sota/image-classification-on-imagenet





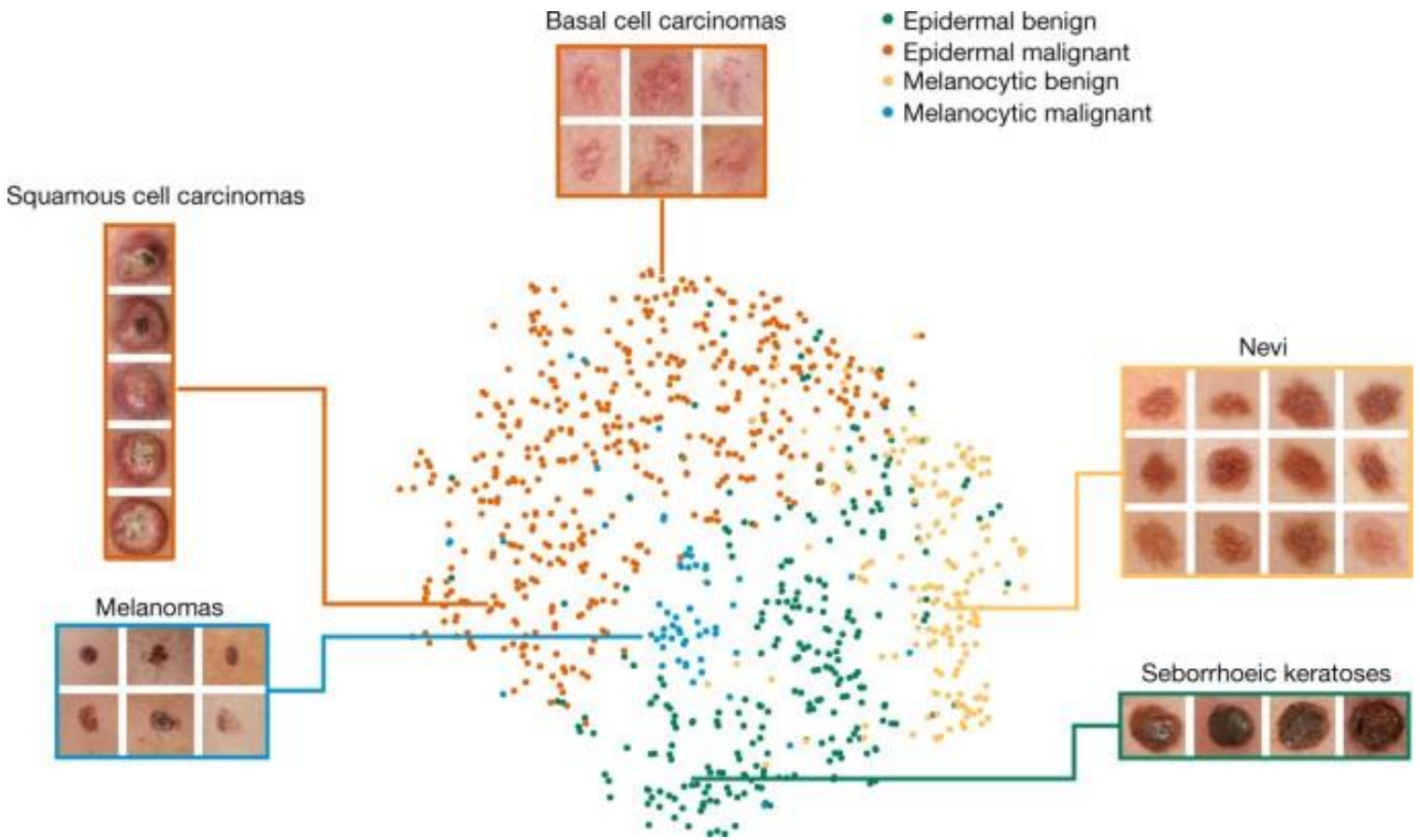
Step 0 → Step 4 → Step 48 → Step 2048

Neural network input optimization: https://distill.pub/2017/feature-visualization/

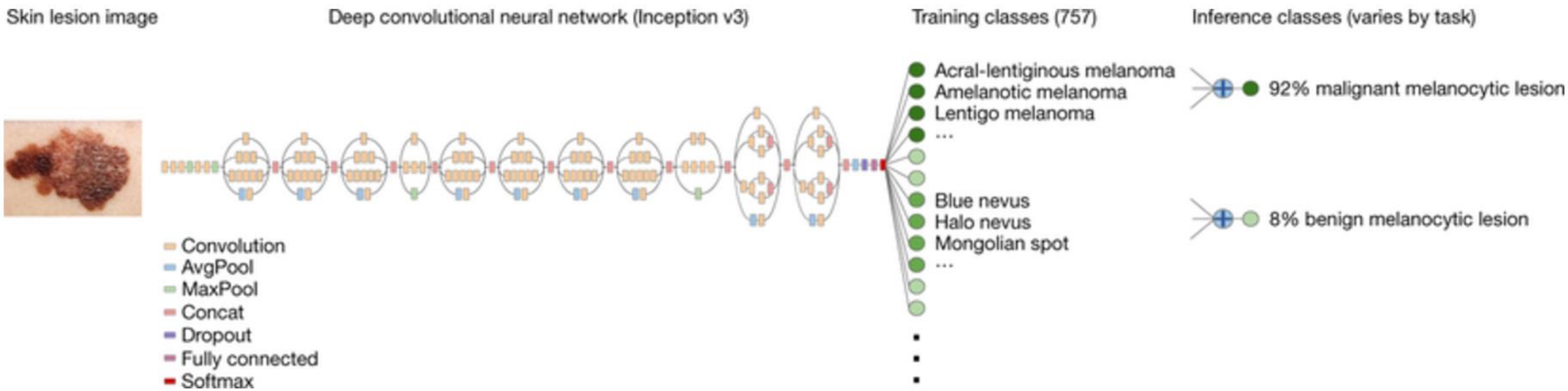


Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

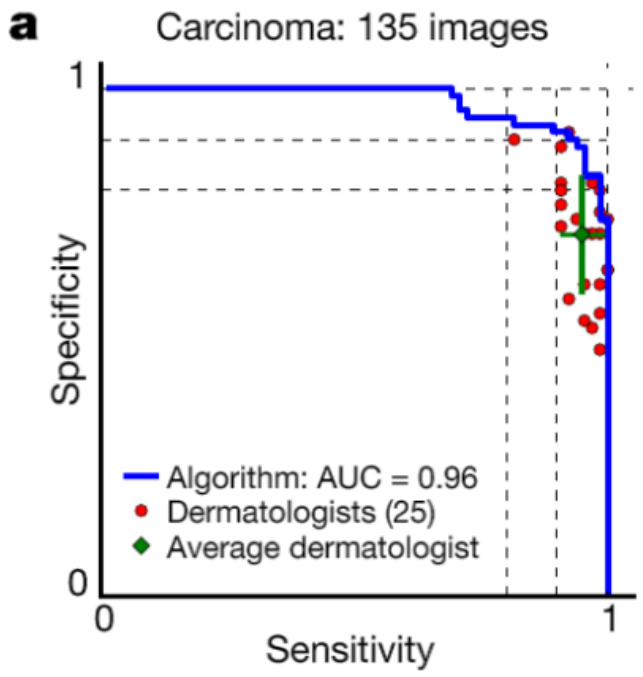
Neural network input optimization: https://distill.pub/2017/feature-visualization/



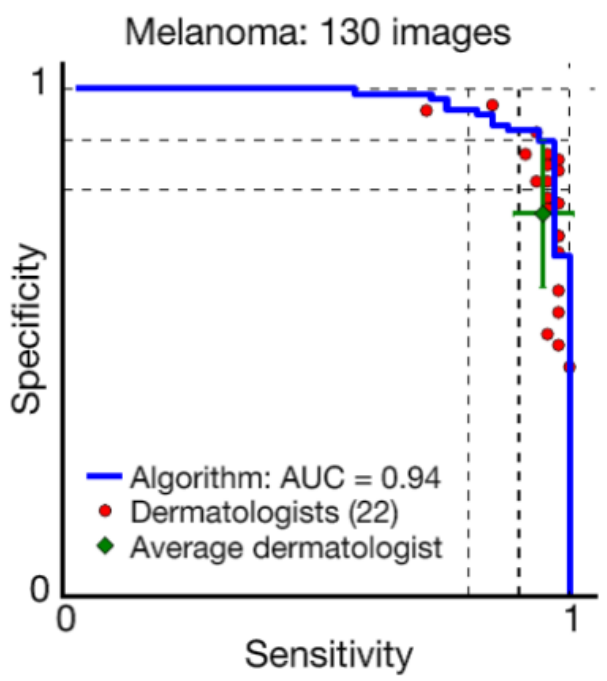
Thrun *et al*., Dermatologist-level classification of skin cancer with deep neural networks, Nature (2017)



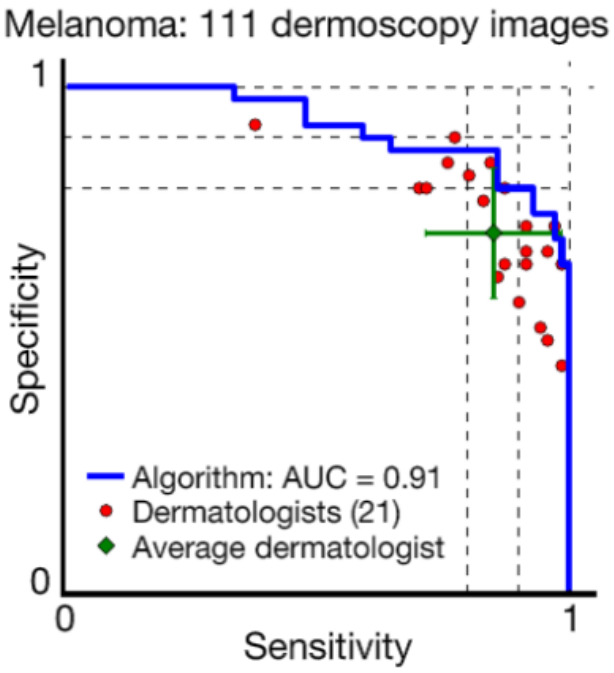
Thrun *et al*., Dermatologist-level classification of skin cancer with deep neural networks, Nature (2017)



keratinocyte carcinomas: 65
benign seborrheic keratoses: 75

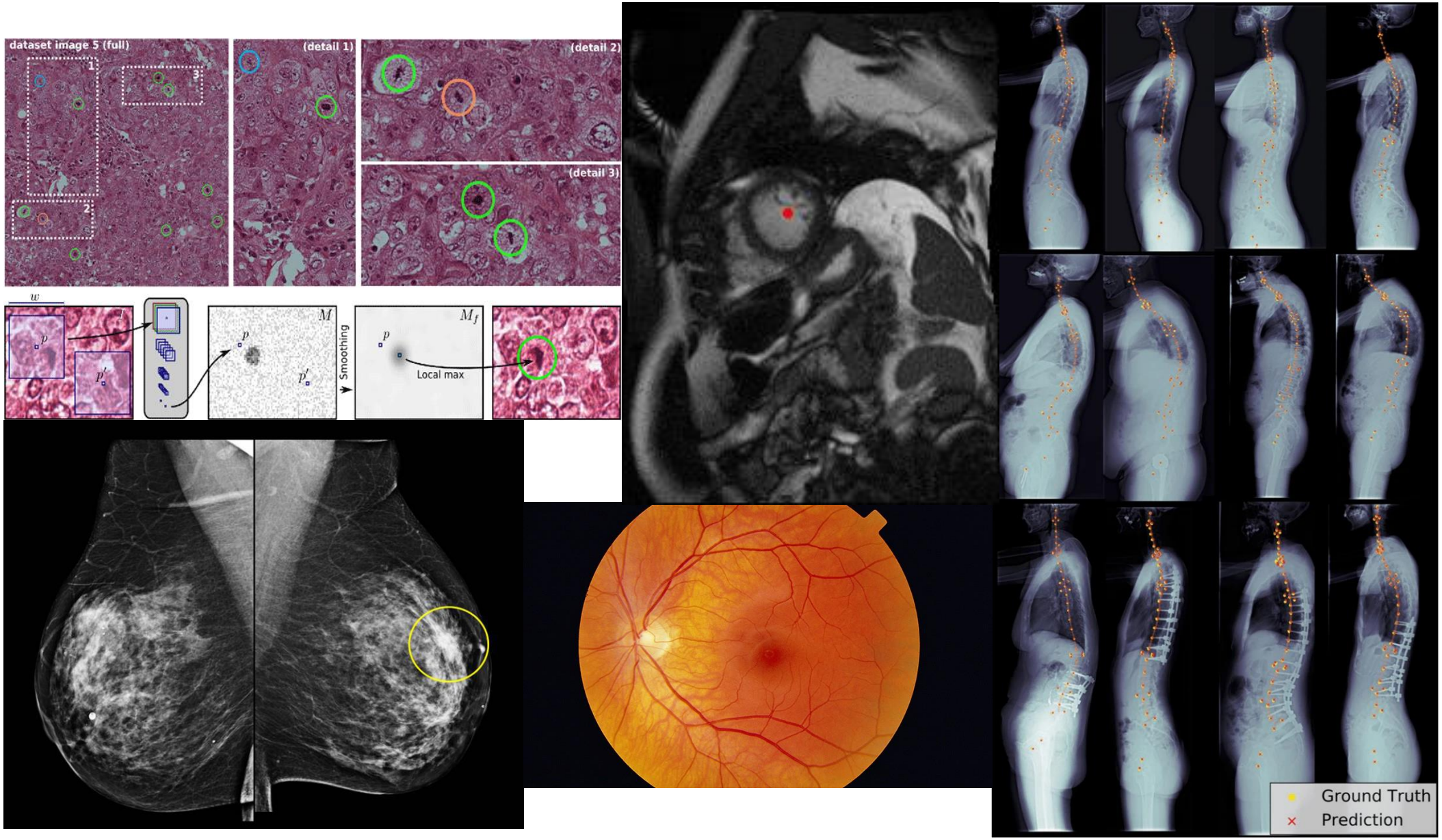


malignant melanomas: 33
benign nevi: 97



malignant melanomas: 71
benign nevi: 40

Thrun *et al.*, Dermatologist-level classification of skin cancer with deep neural networks, Nature (2017)

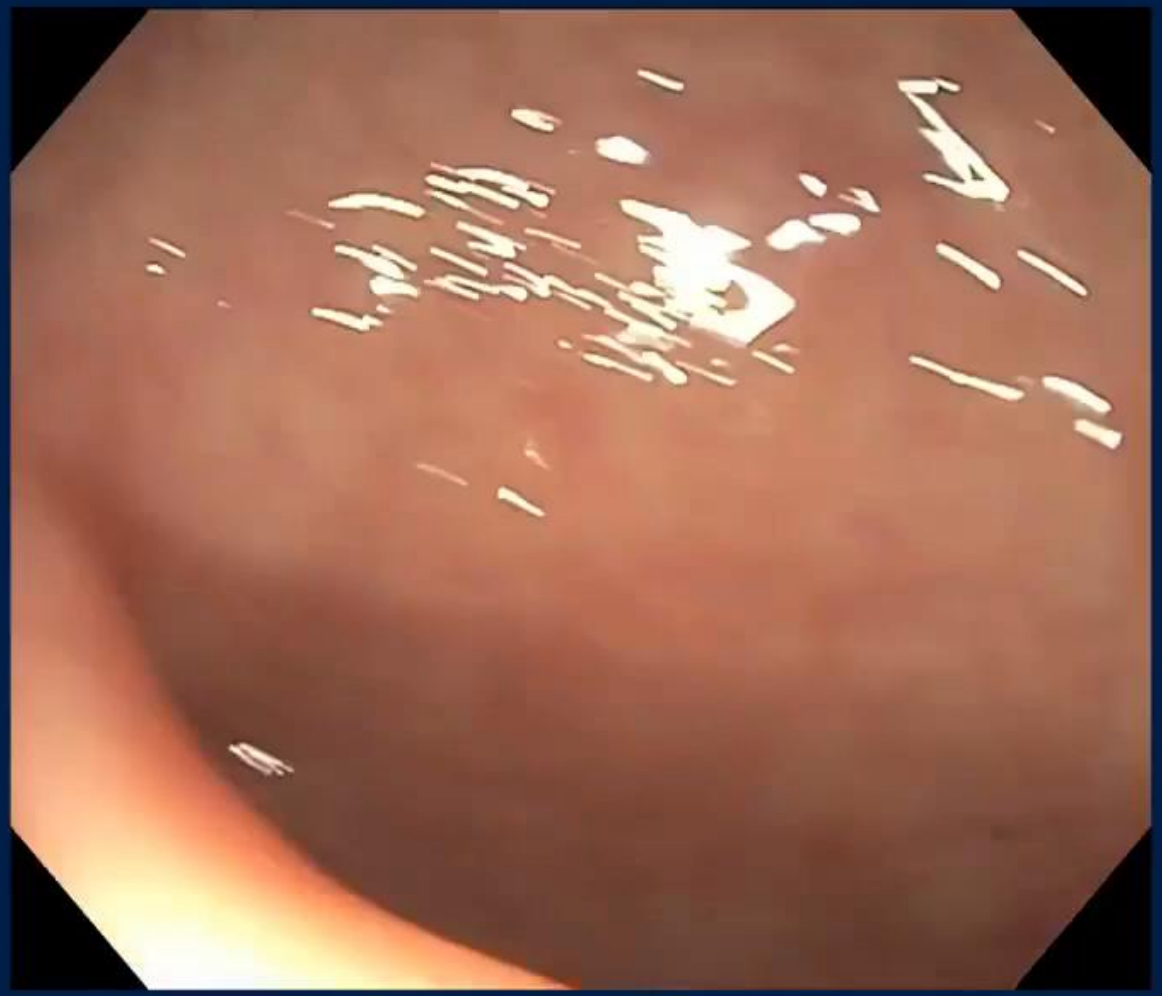


http://people.idsia.ch/~juergen/deeplearningwinsMICCAIgrandchallenge.html

kaggle.com/second-annual-data-science-bowl (2015)

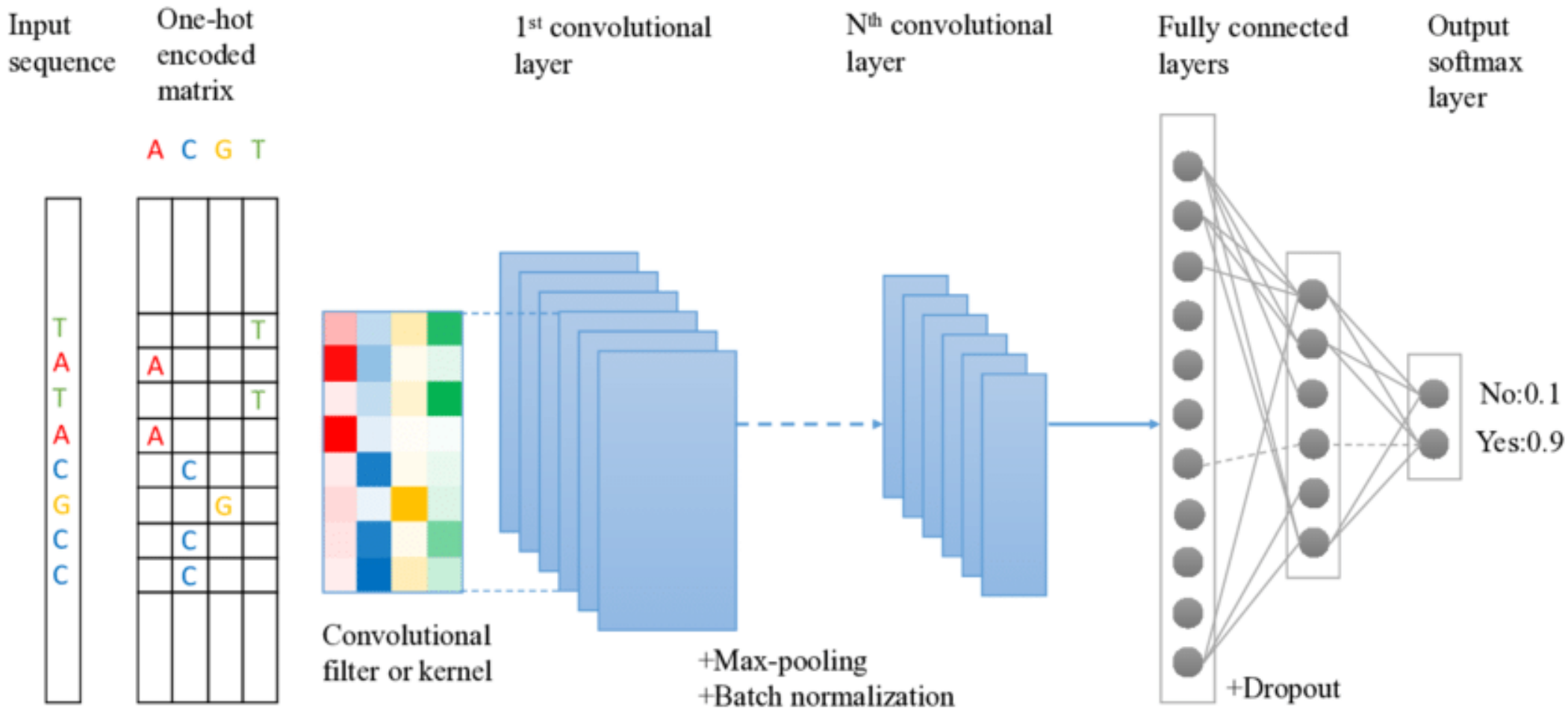
Yeh, YC. *et al.* Deep learning approach for automatic landmark detection and alignment analysis in whole-spine lateral radiographs. *Sci Rep* **11,** 7618 (2021)

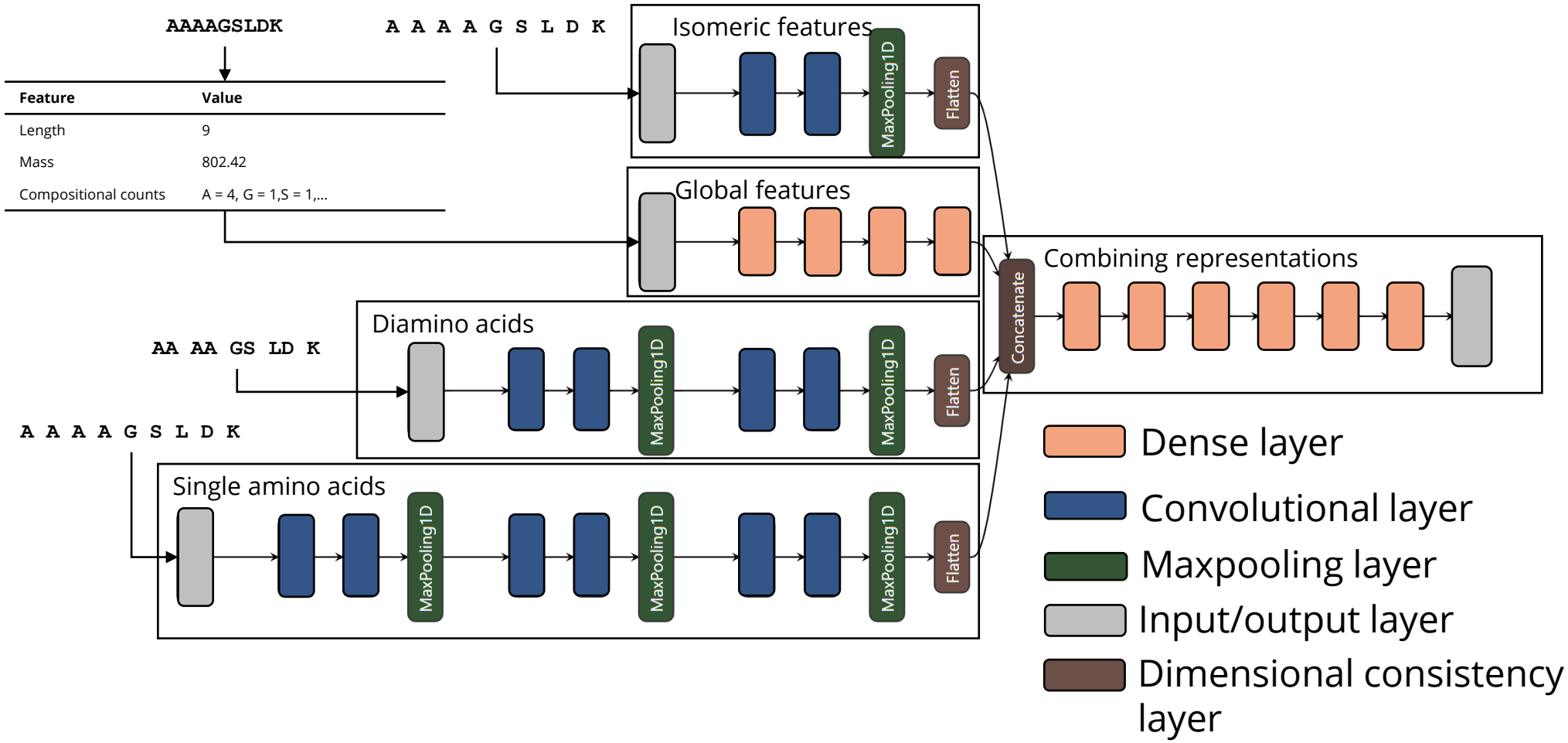
GI Genius™ Intelligent Endoscopy Module



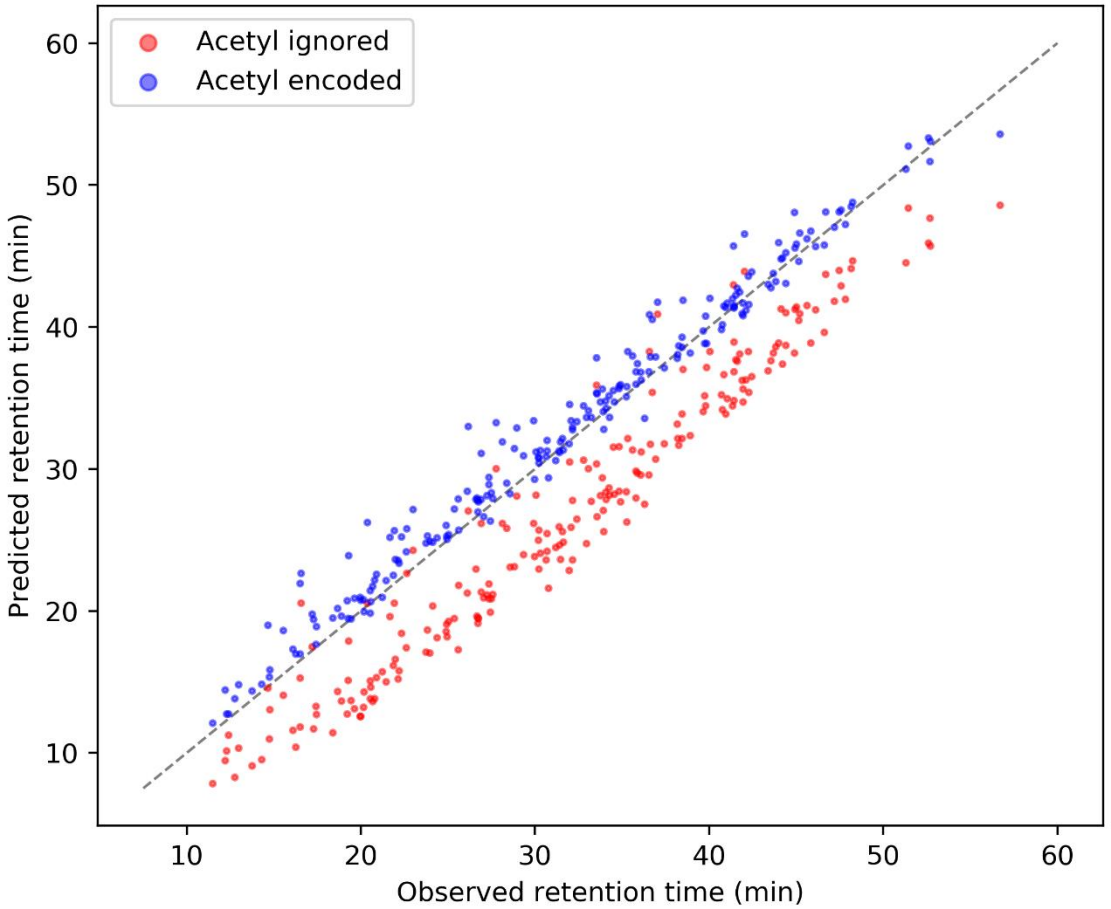
Artificial Intelligence + Colonoscopy

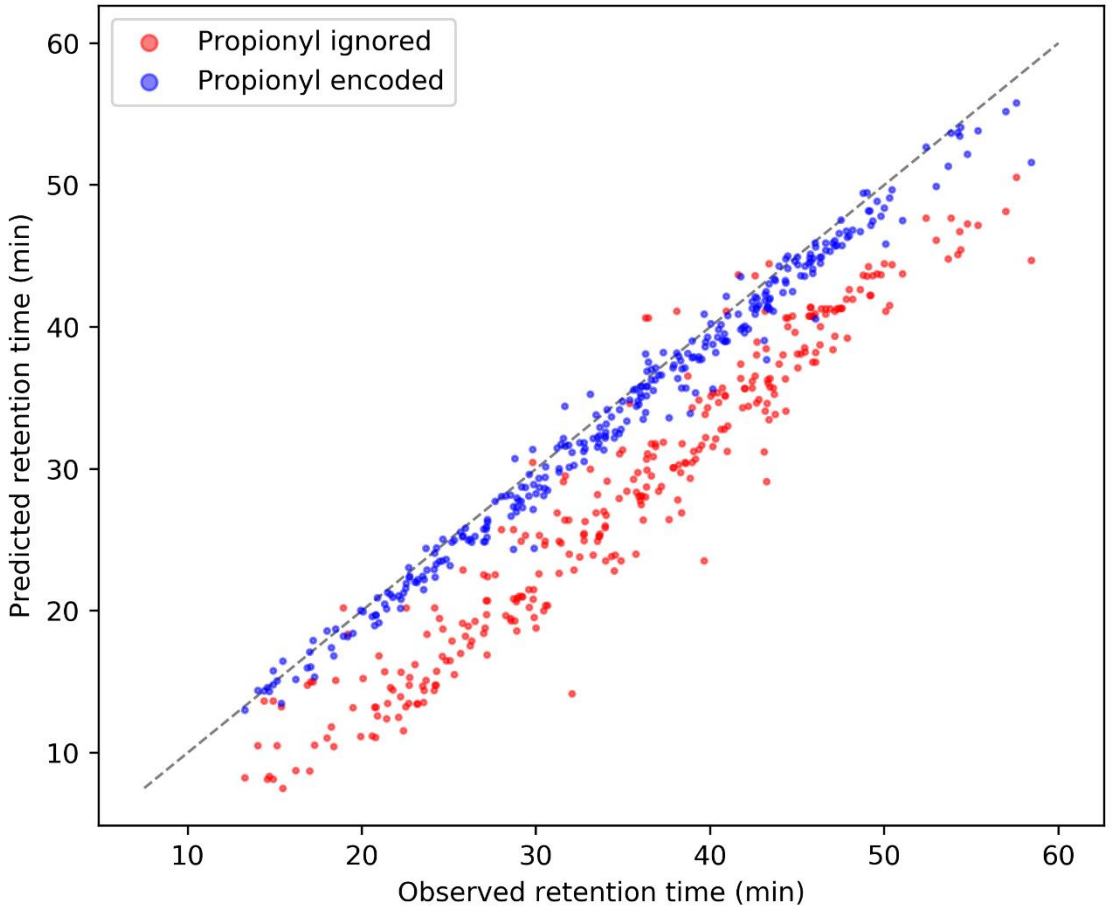
Medtronic's GI Genius


Input sequence
One-hot encoded matrix
A C G T
T A T A C G C C
1st convolutional layer
Nth convolutional layer
Fully connected layers
Output softmax layer
Convolutional filter or kernel
+Max-pooling
+Batch normalization
+Dropout
No:0.1
Yes:0.9

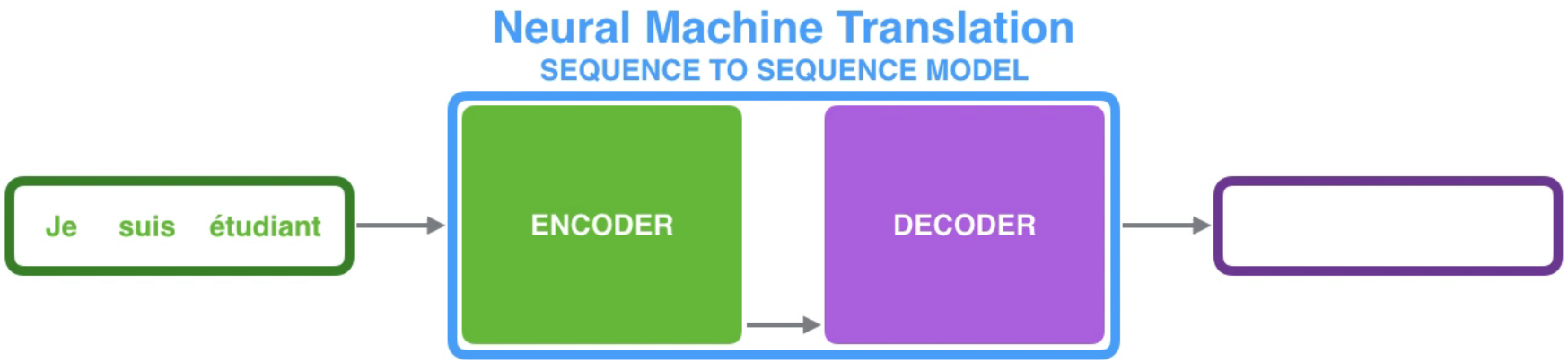


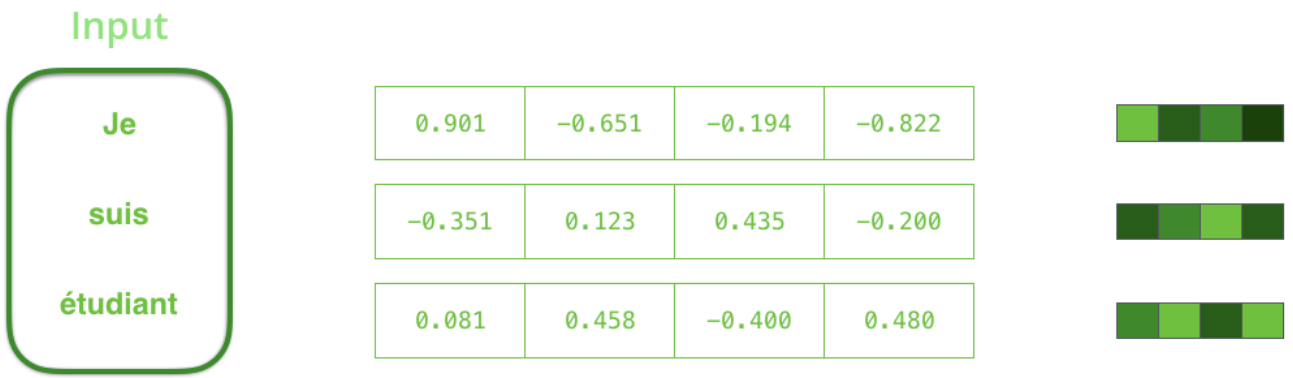
Bouwmeester, R. *et al.* DeepLC can predict retention times for peptides that carry as-yet unseen modifications. *Nat Methods* **18,** 1363–1369 (2021)



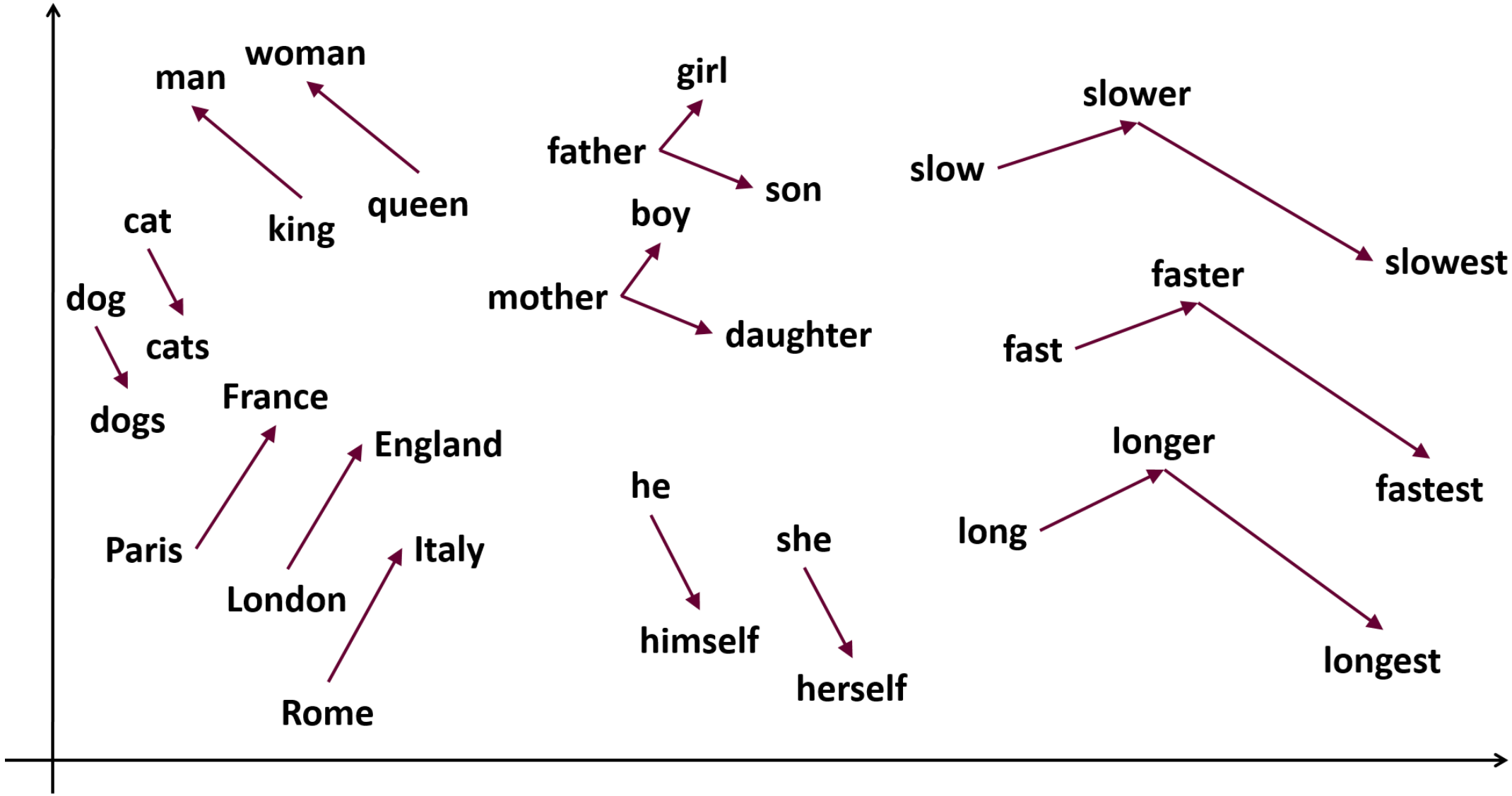


Bouwmeester, R. *et al.* DeepLC can predict retention times for peptides that carry as-yet unseen modifications. *Nat Methods* **18,** 1363–1369 (2021)





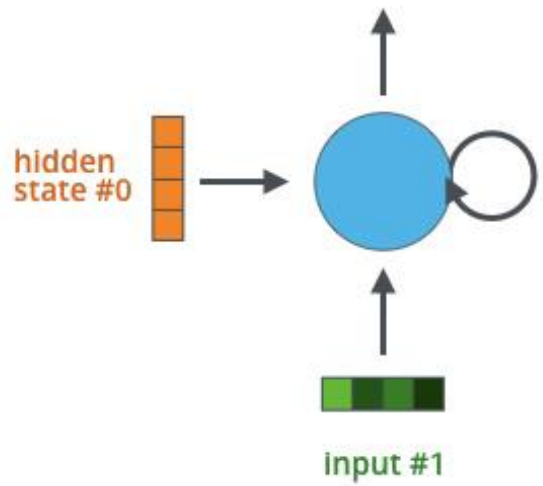
Recurrent neural network (RNN): https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/



Mikolov *et al*. Proceedings of the International Conference on Learning Representations (2013)

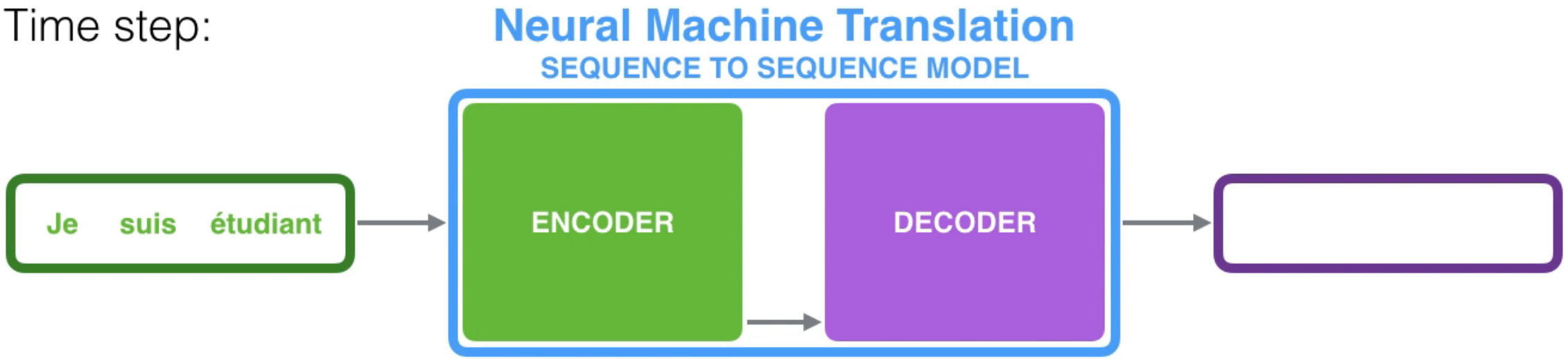
Recurrent Neural Network

Time step #1:
An RNN takes two input vectors:



Recurrent neural network (RNN): https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

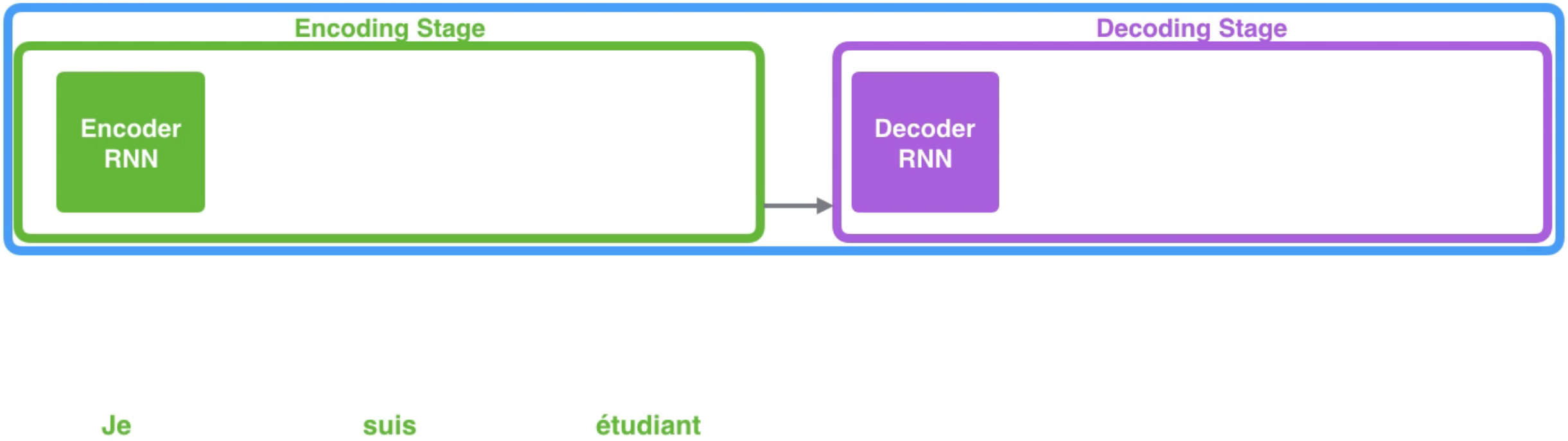
Time step:



Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Neural Machine Translation

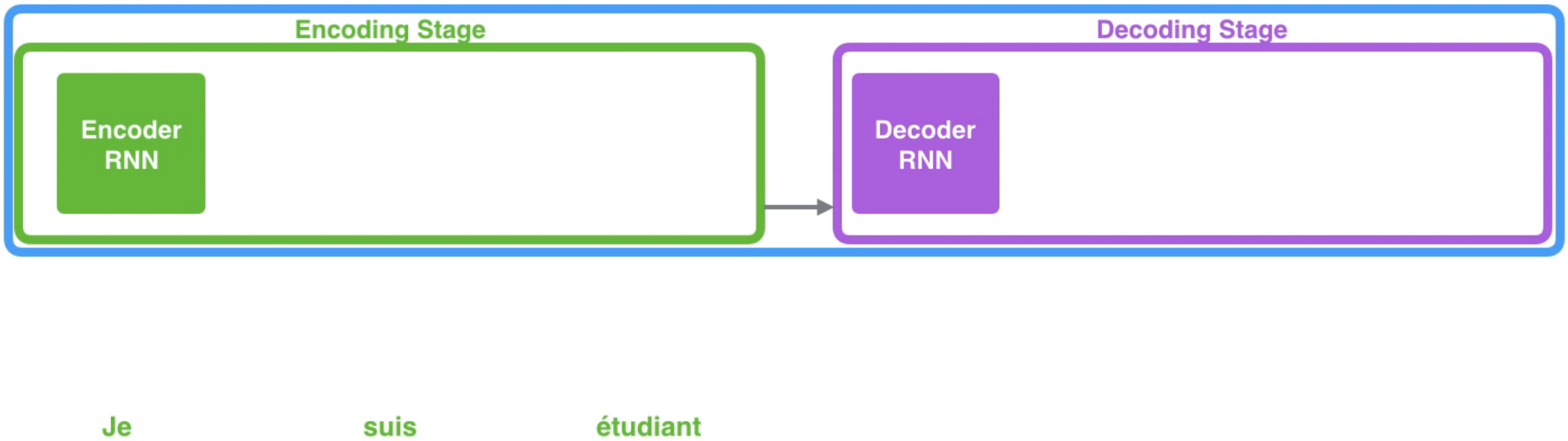
SEQUENCE TO SEQUENCE MODEL



Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL



Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Neural Machine Translation

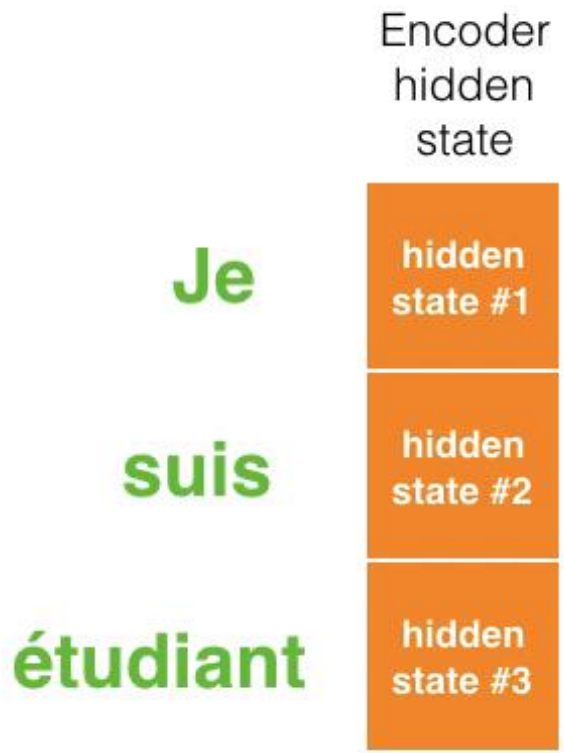
SEQUENCE TO SEQUENCE MODEL WITH ATTENTION

Je suis étudiant

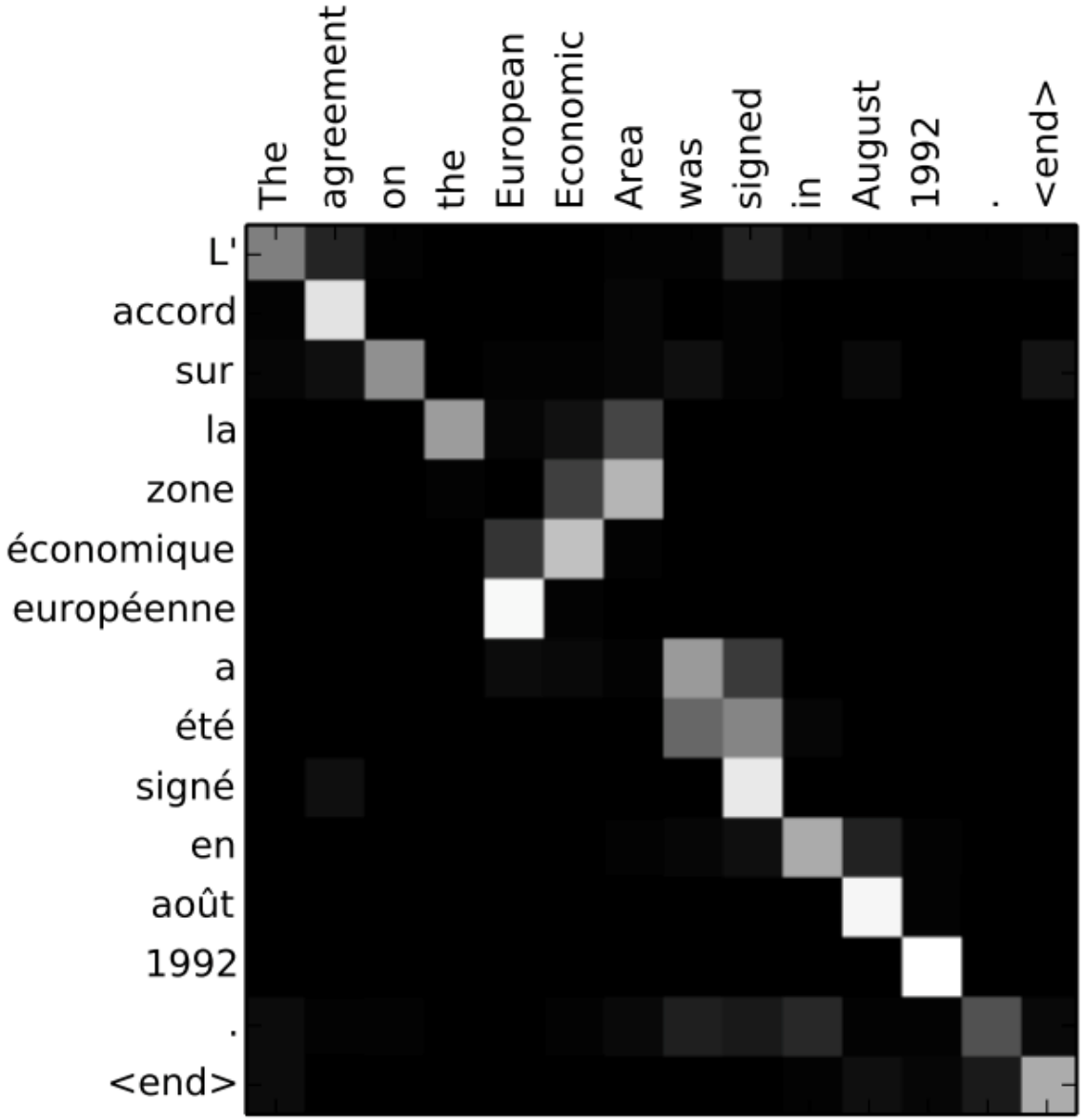
Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Attention at time step 4

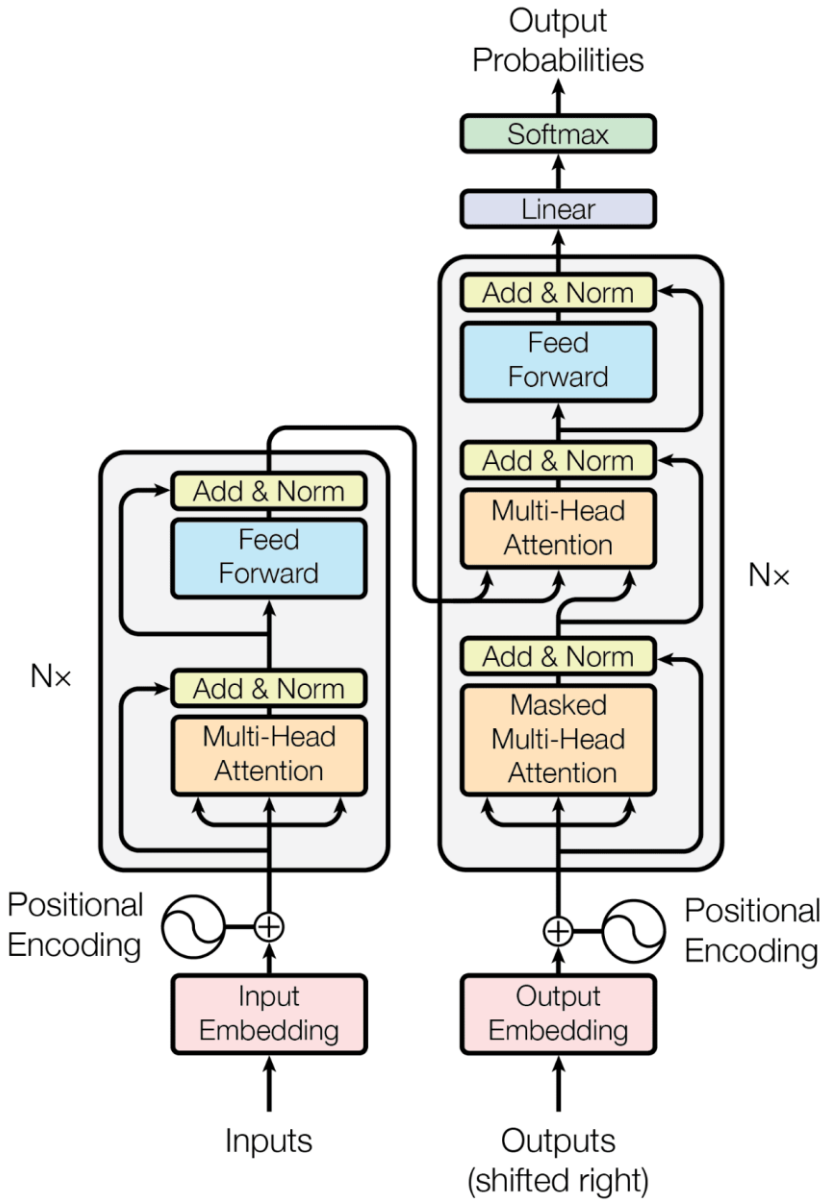
Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/



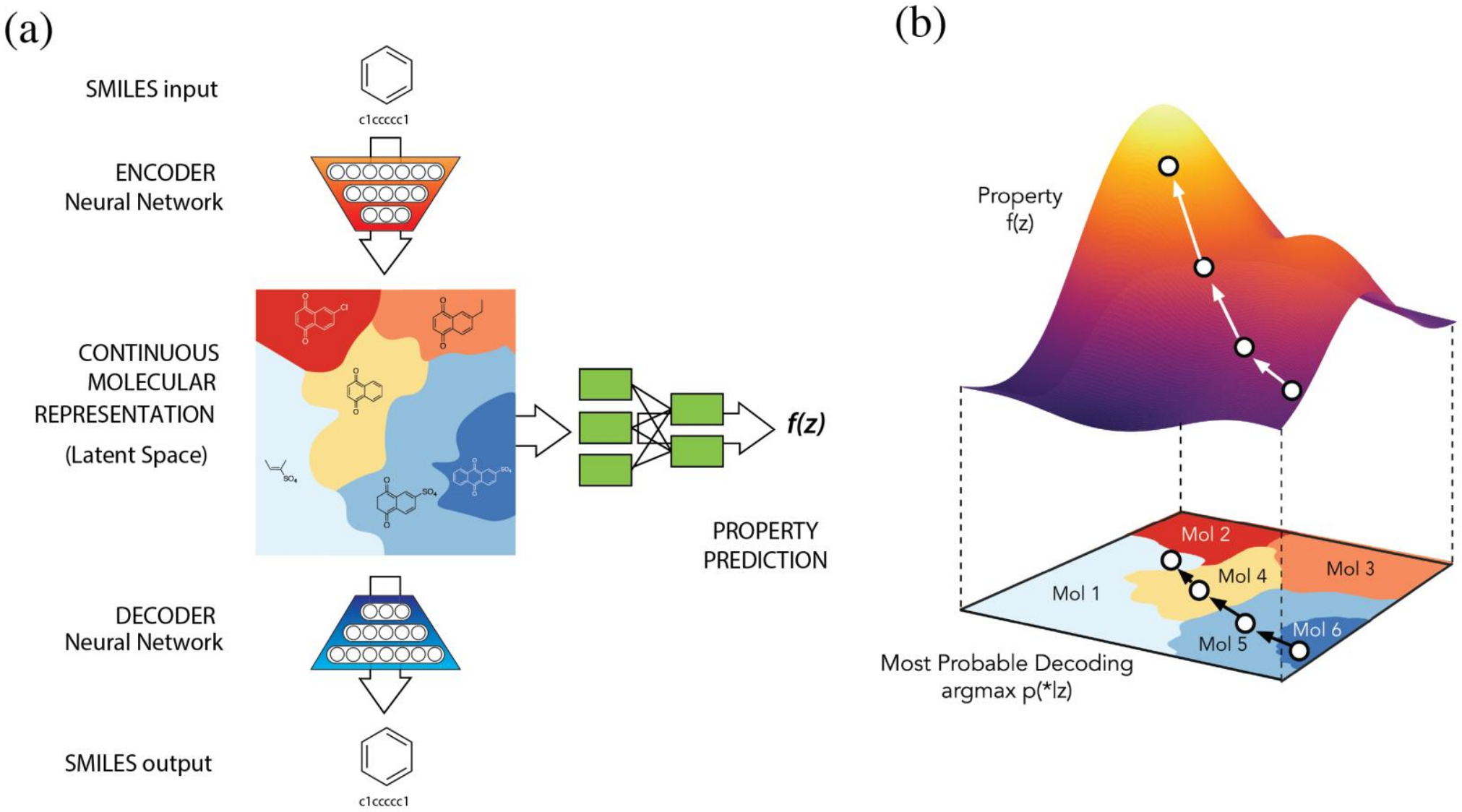
Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/



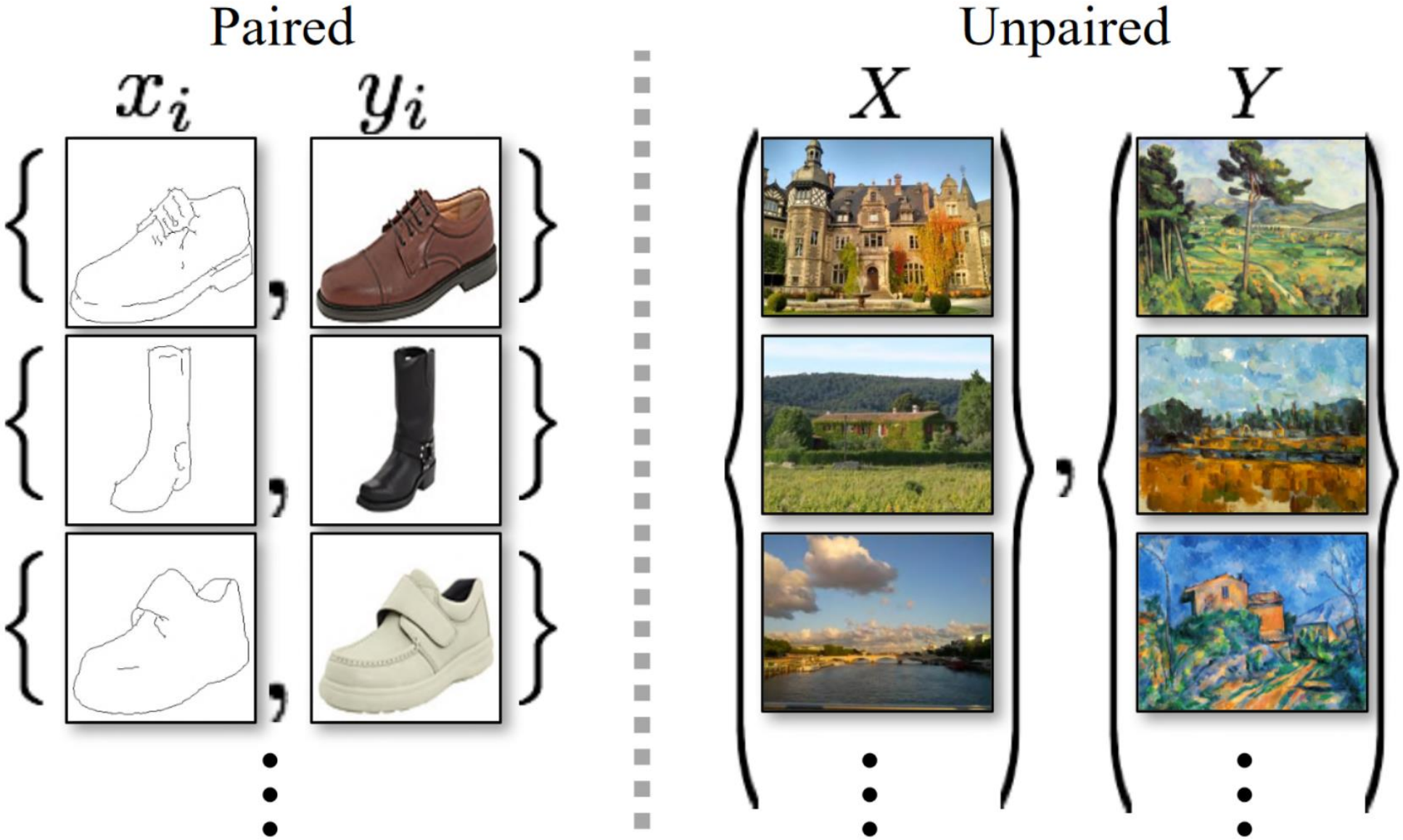
Encoder-decoder architecture: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/



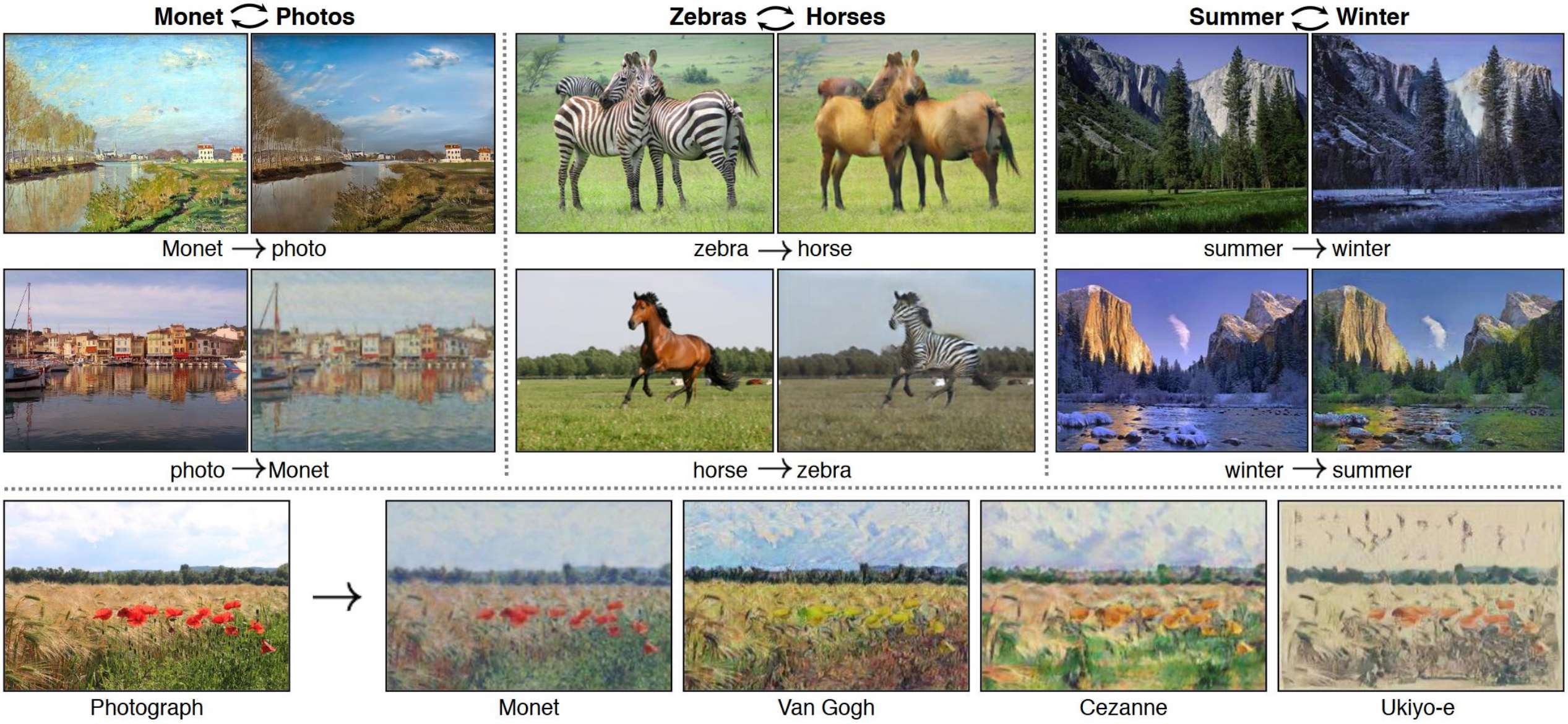
Ashish Vaswani et al., Attention is all you need. (NIPS'17)
Transformer architecture: https://jalammar.github.io/illustrated-transformer/



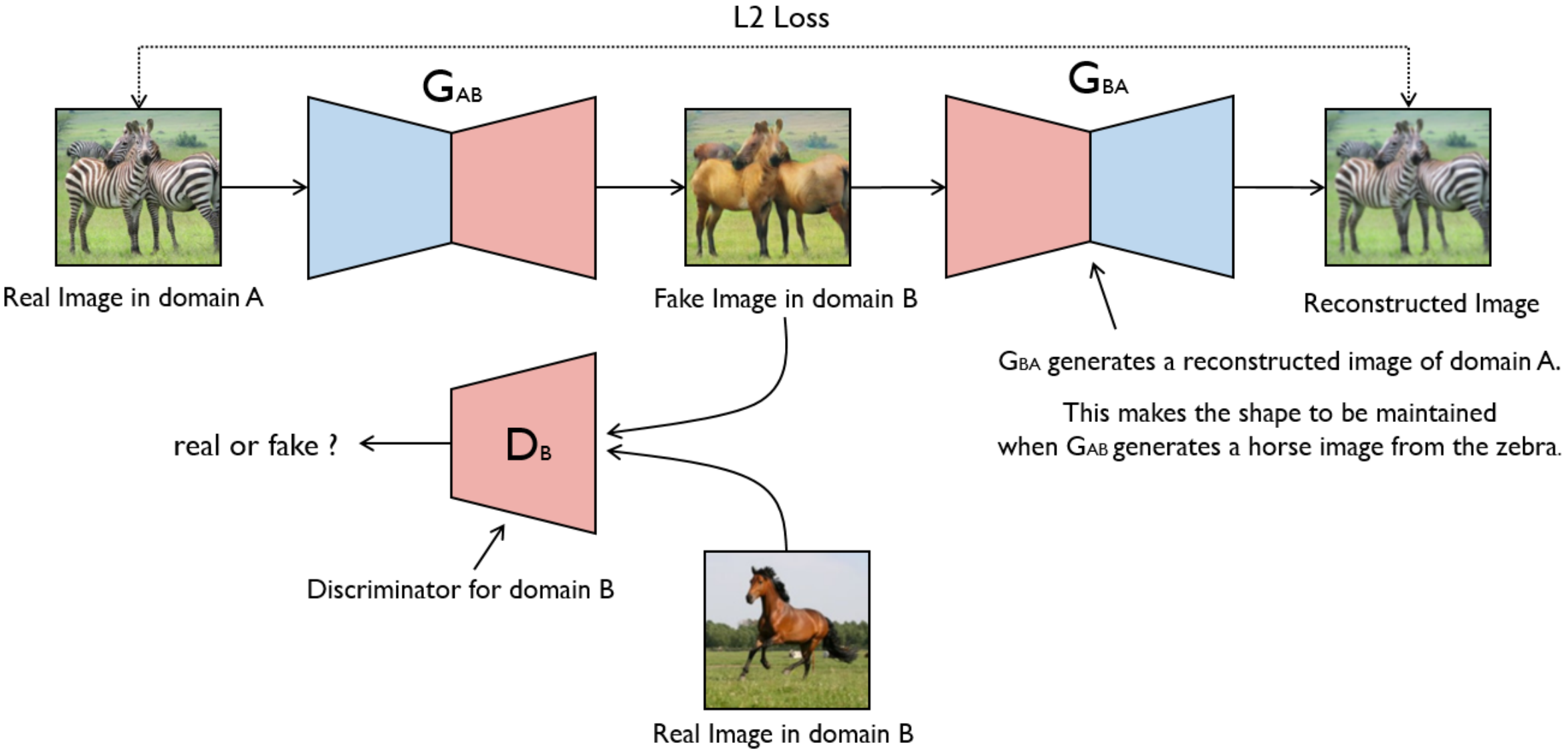
Auto-encoders: Gómez-Bombarelli *et al.* ACS Central Science (2019)



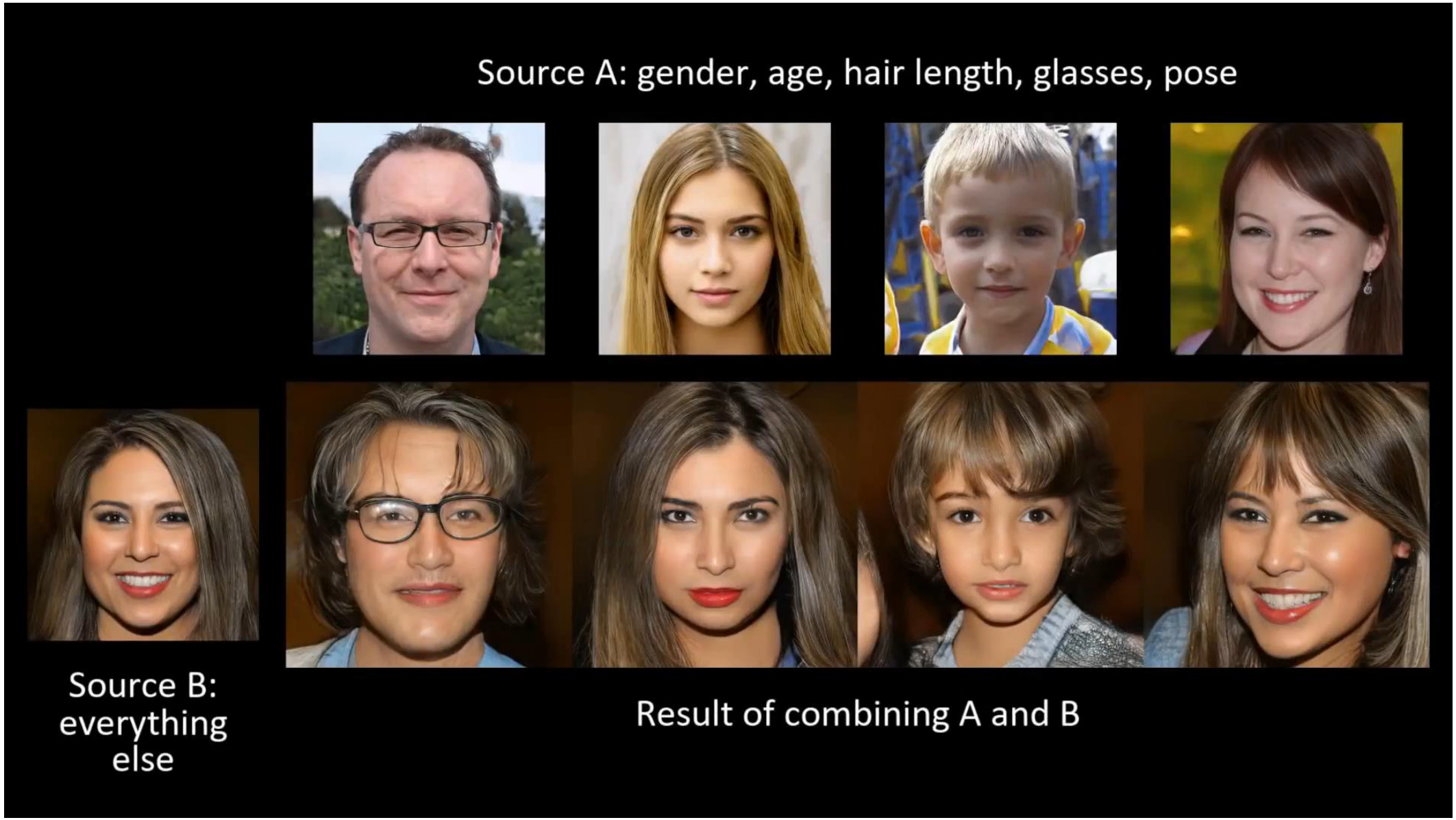
un-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV (2017)



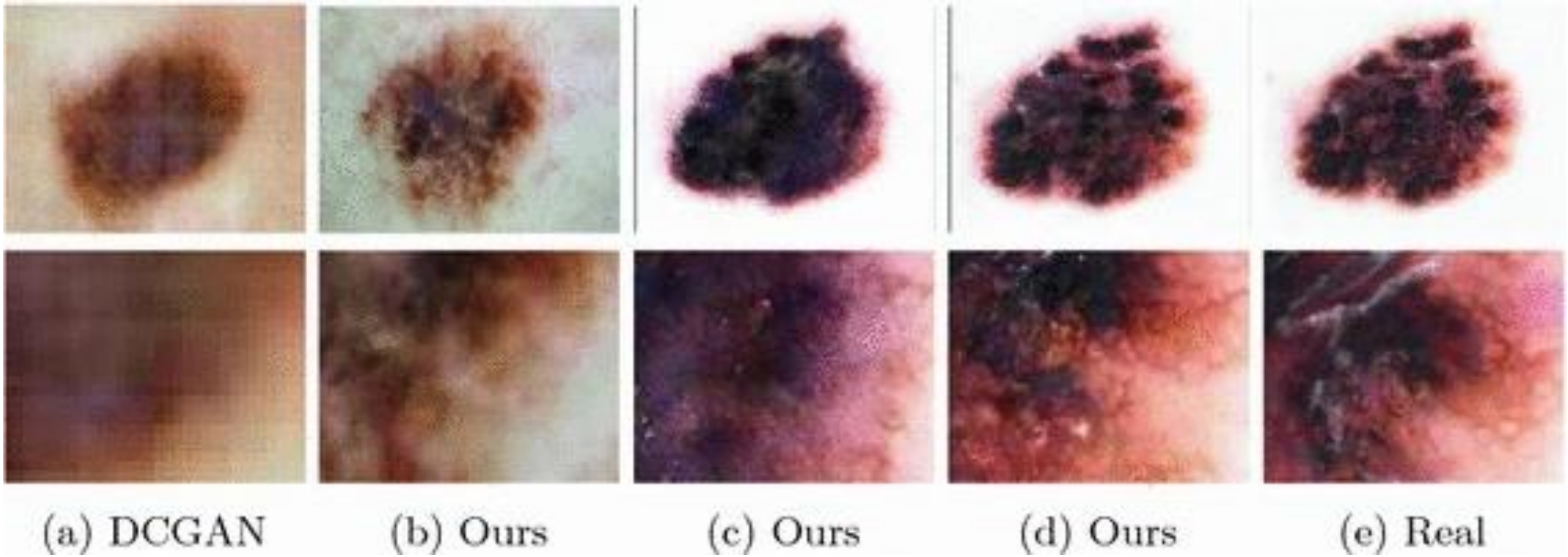
un-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV (2017)



un-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV (2017)

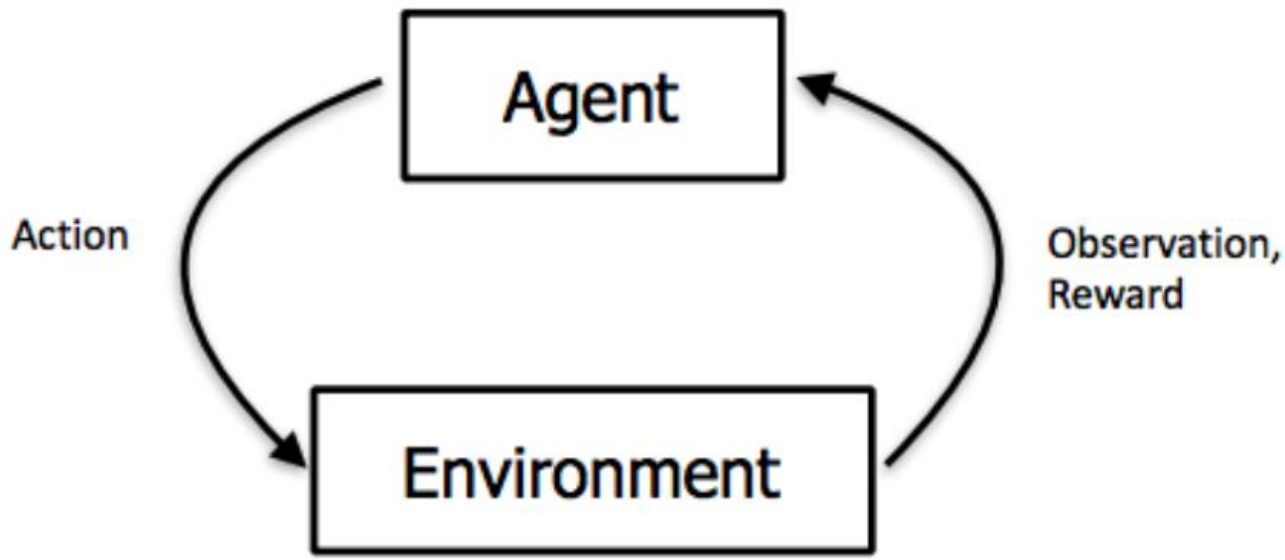


Generative adversarial networks: Karras *et al*. IEEE Conference on Computer Vision and Pattern Recognition (2019)



Bissoto, A. *et al*., Skin Lesion Synthesis with Generative Adversarial Networks. Lecture Notes in Computer Science (2018)

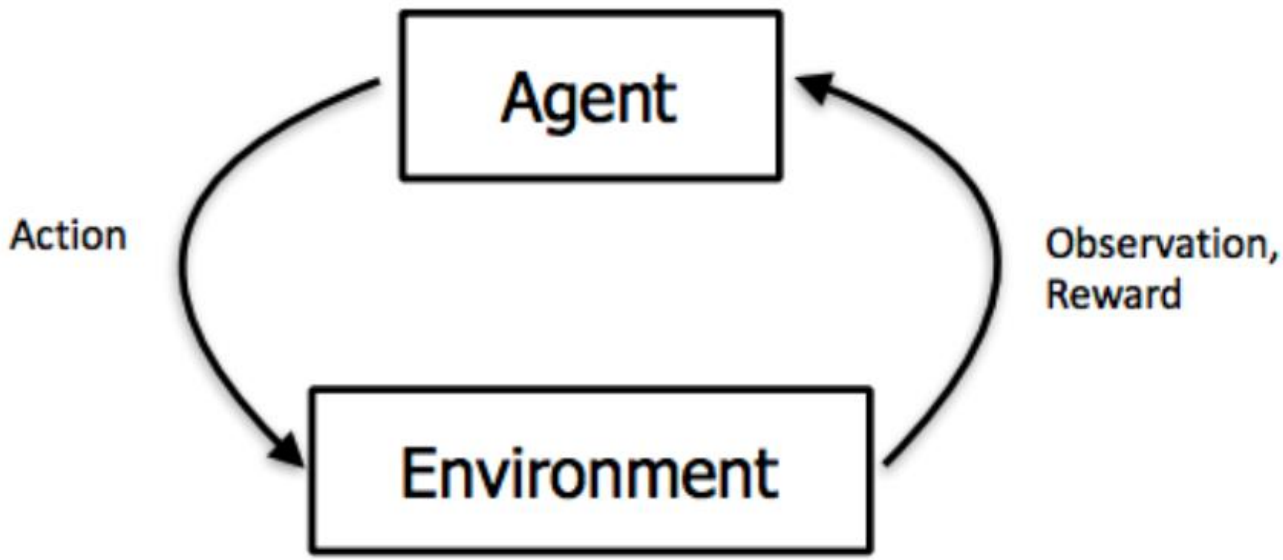
https://openai.com/dall-e-2/#demos



The agent **observes** the environment, takes an **action** to interact with the environment, and receives positive or negative **reward.** Diagram from Berkeley's CS 294: Deep Reinforcement Learning by John Schulman & Pieter Abbeel

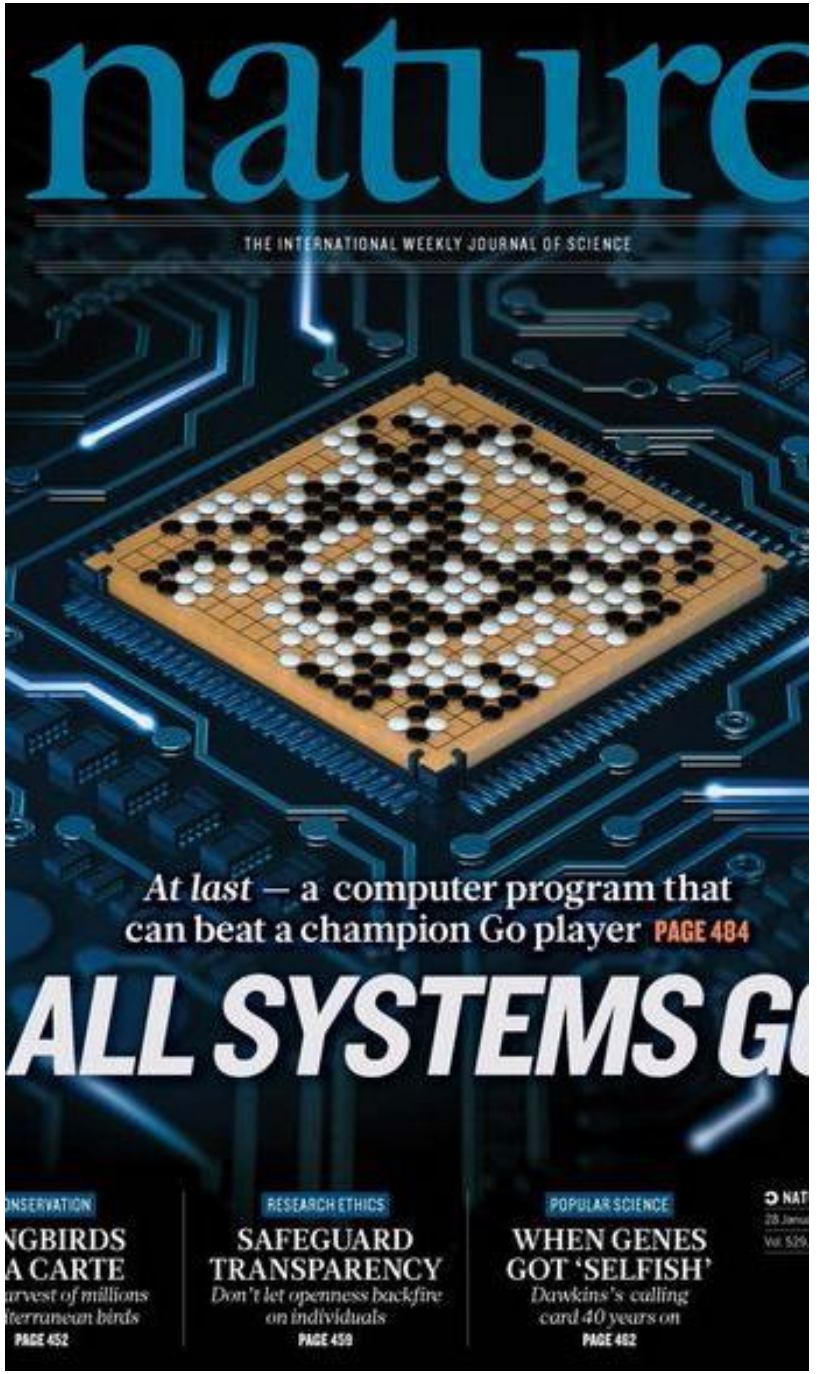
$$a = \pi(s)$$

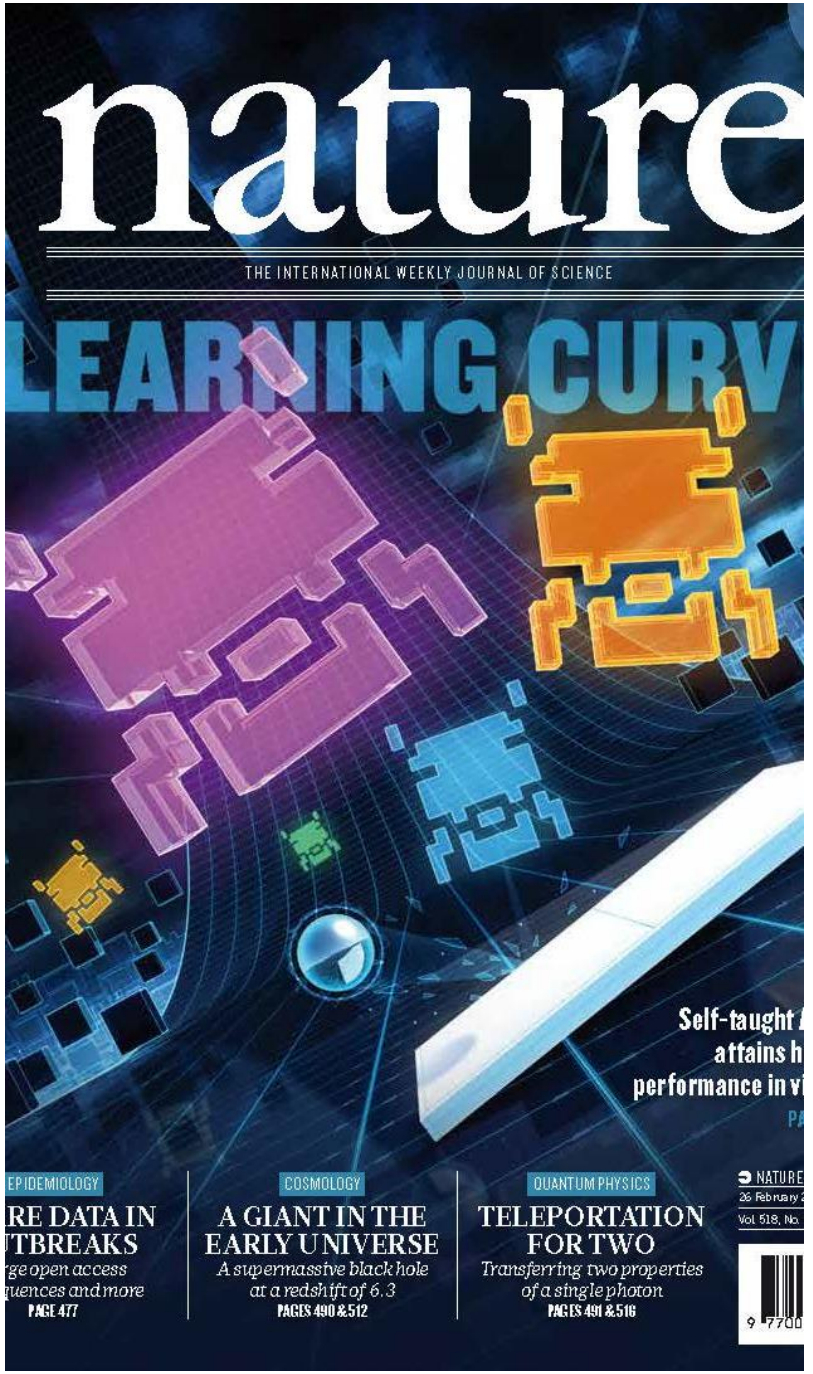
A **policy** is a map from **state** to **action.**

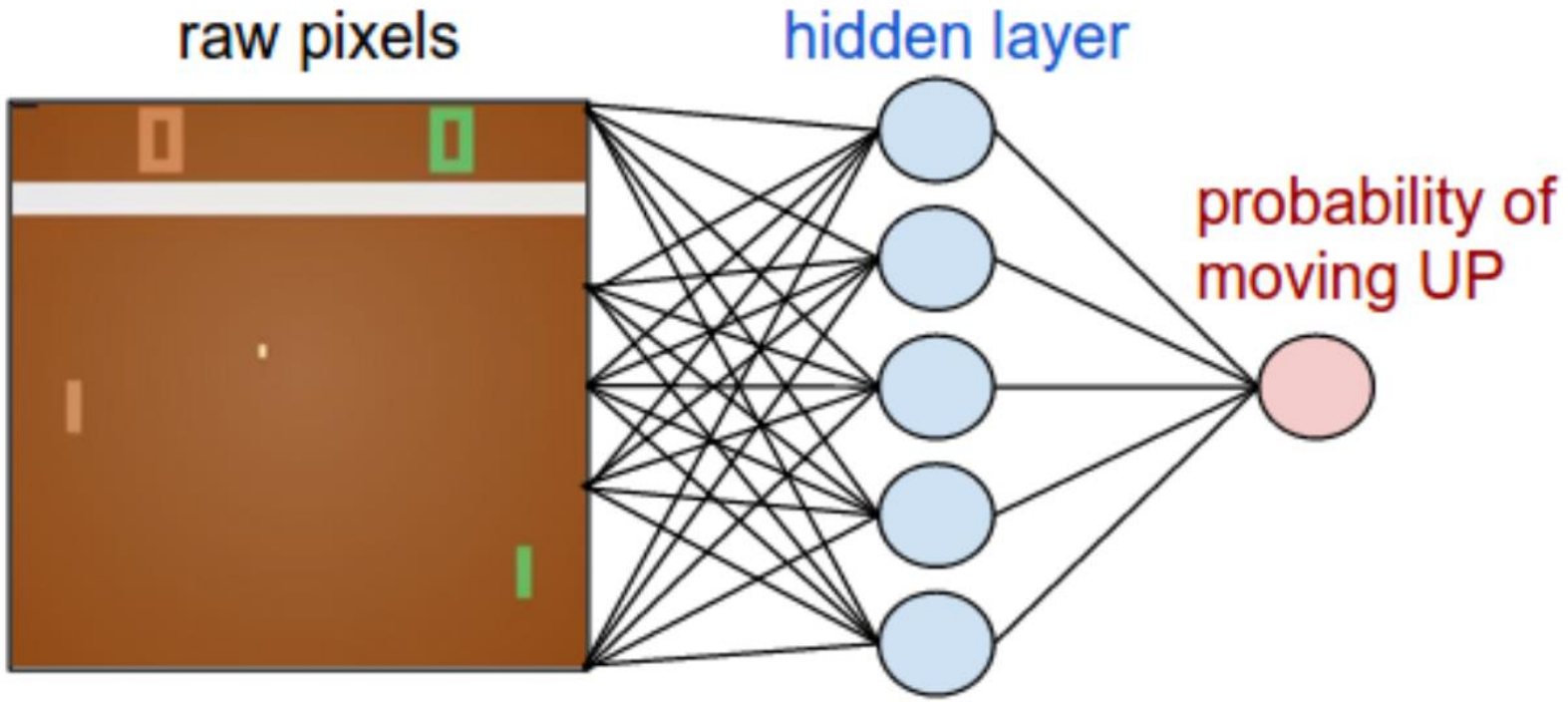


The agent **observes** the environment, takes an **action** to interact with the environment, and receives positive or negative **reward.** Diagram from Berkeley's CS 294: Deep Reinforcement Learning by John Schulman & Pieter Abbeel


mperspedi audicto p. XX
Quasi volut accae cullit ad queomni quis pp. XX & XX
Perspedi cuptatur au volut accae cullit p. X
cience
$15
7 DECEMBER
sciencemag.org
AAAS
31.63%
18.47%
A DIGITAL
PRODIGY
AlphaZero teaches itself chess, shogi, and Go
pp. XXXX, XXXX, & XXXX

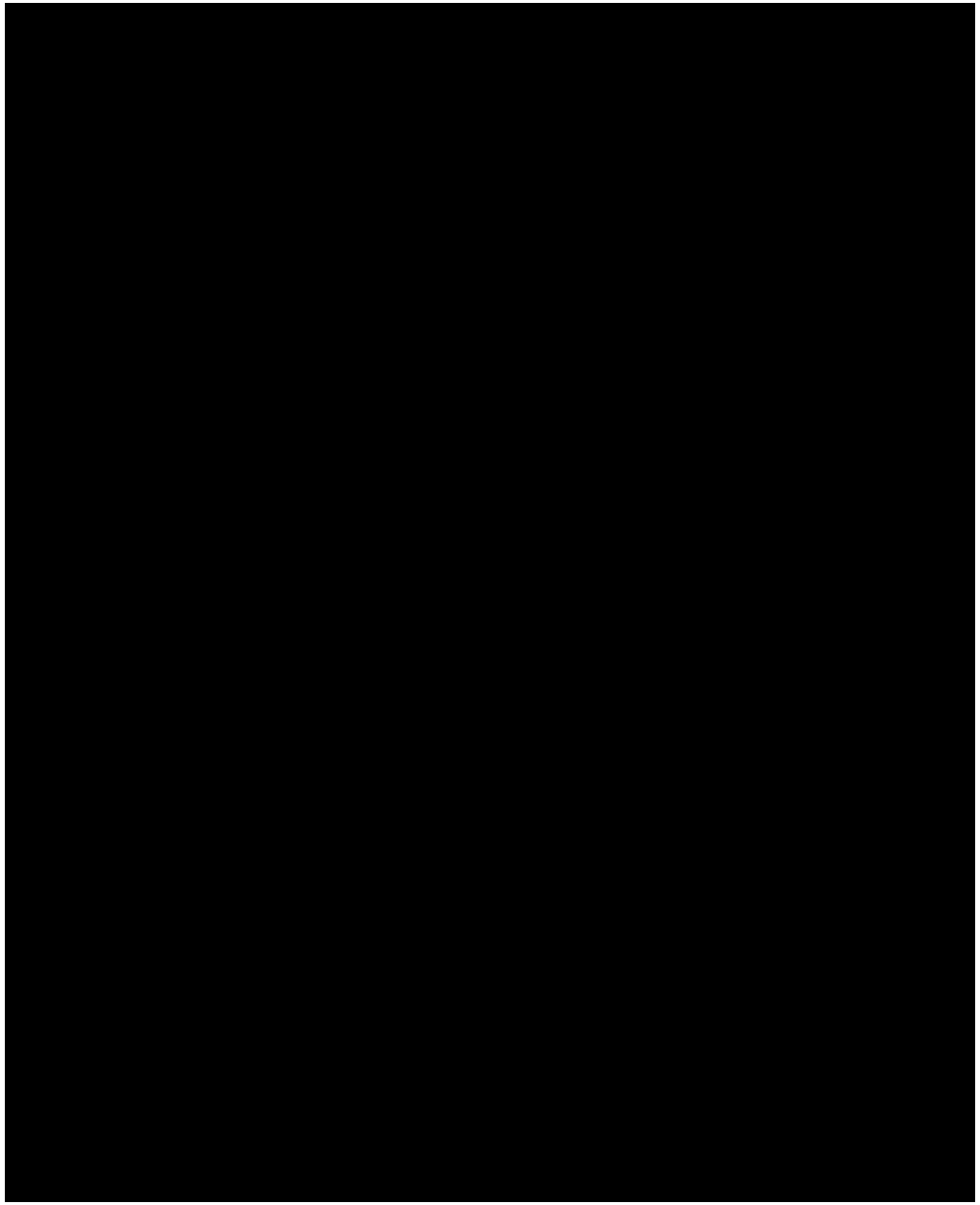

nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS G
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURV
Self-taught
attains h
performance in v
COSMOLOGY
A GIANT IN THE EARLY UNIVERSE
A supermassive black hole at a redshift of 6.3
PAGES 490 & 512
QUANTUM PHYSICS
TELEPORTATION FOR TWO
Transferring two properties of a single photon
PAGES 491 & 516



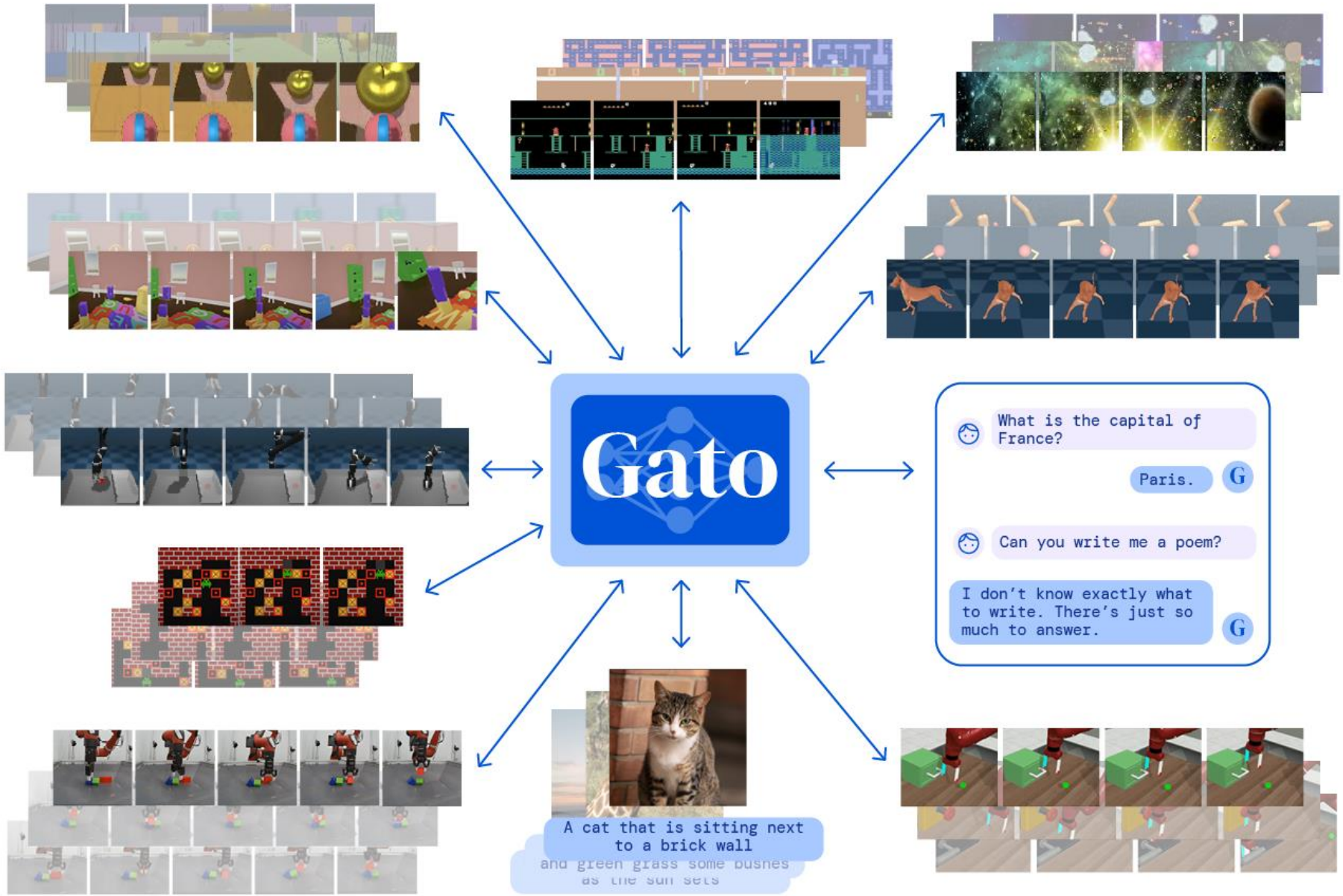
In a policy gradient network, the agent learns the optimal policy by adjusting its weights through gradient descent based on reward signals from the environment. Image via http://karpathy.github.io/2016/05/31/rl/

Mnih, Volodymyr *et al*., Playing Atari with Deep Reinforcement Learning (2013)



Mnih, Volodymyr *et al*., Playing Atari with Deep Reinforcement Learning (2013)

https://openai.com/blog/emergent-tool-use/



https://www.deepmind.com/publications/a-generalist-agent