3011979 Practical Python for Data Sciences and Machine Learning

L13 Introduction to deep learning

Apr 22th, 2022

Sira Sriswasdi, Ph.D.
Research Affairs, Faculty of Medicine
Chulalongkorn University

Multi-layer Perceptron

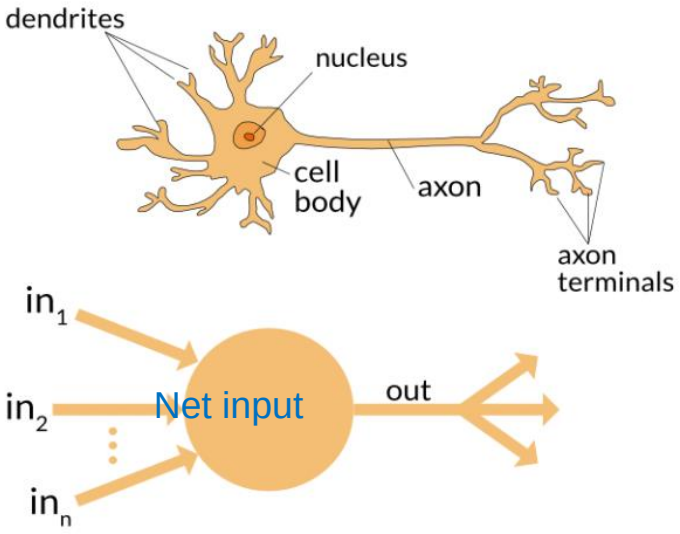
Fully Connected Neural Network

Dense Neural Network

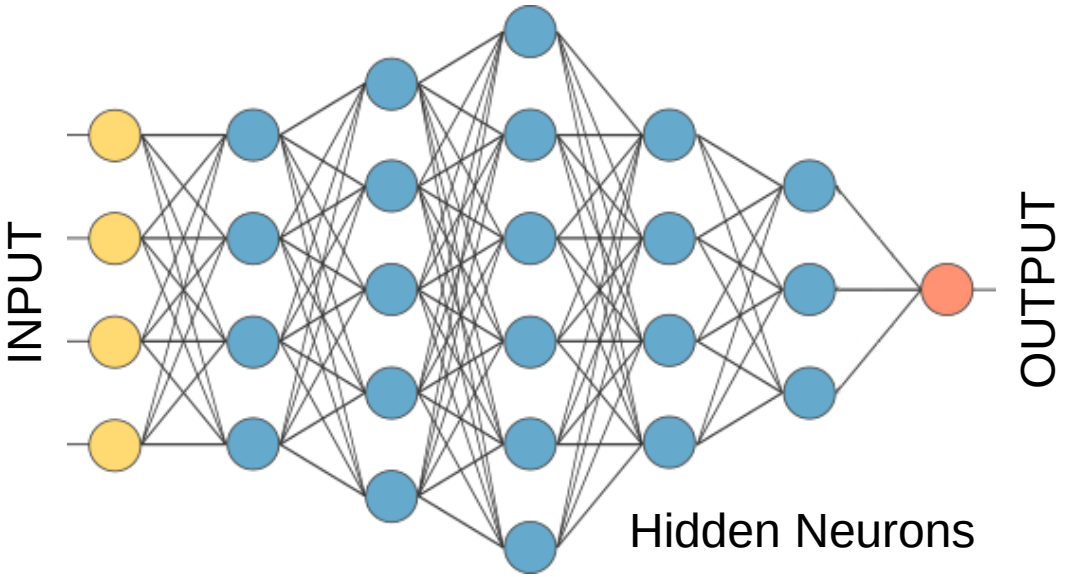
Feed-forward Neural Network

Multi-layer perceptron

Neuron



Artificial Neural Network



Source: www.decom.ufop.br/imobilis/fundamentos-de-redes-neurais/

- MLP is the basic artificial neural network architecture

Artificial neuron

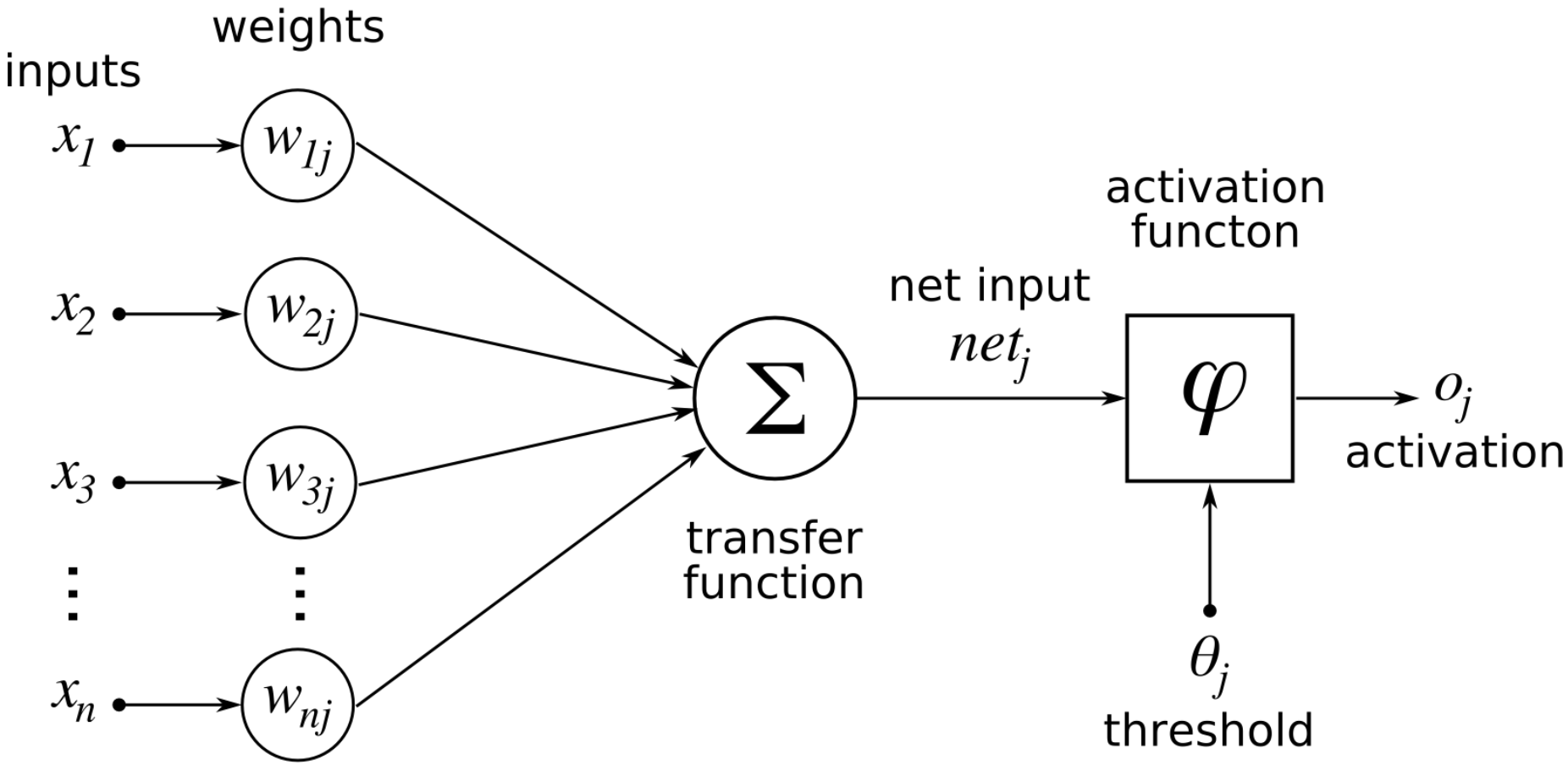
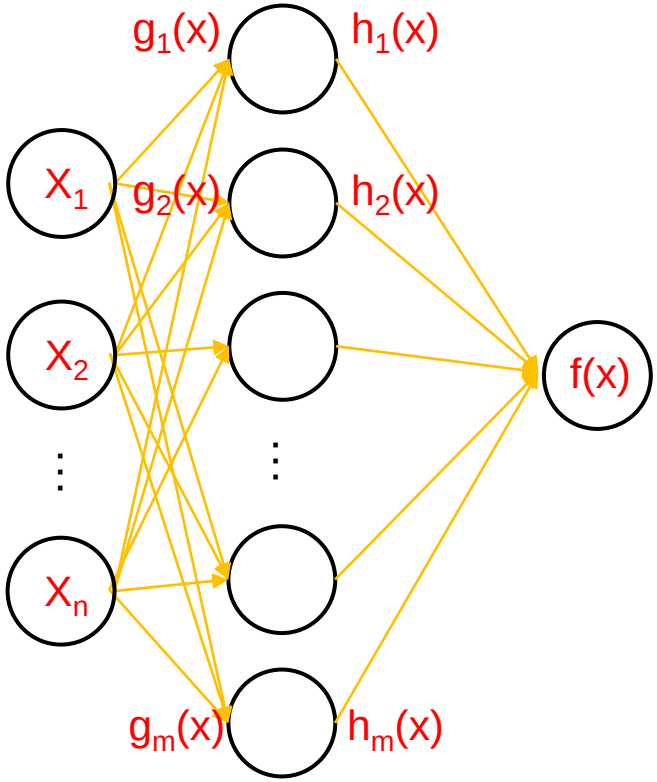


Image from Wikipedia.com

Calculation inside MLP



Neuron input = linear combination

$$g_1(x) = w_{1,1}x_1 + \cdots + w_{1,n}x_n$$

…

$$g_m(x) = w_{m,1}x_1 + \cdots + w_{m,n}x_n$$

Neuron output = activation function

$$h_1(x) = \frac{1}{1 + e^{-g_1(x)}}$$

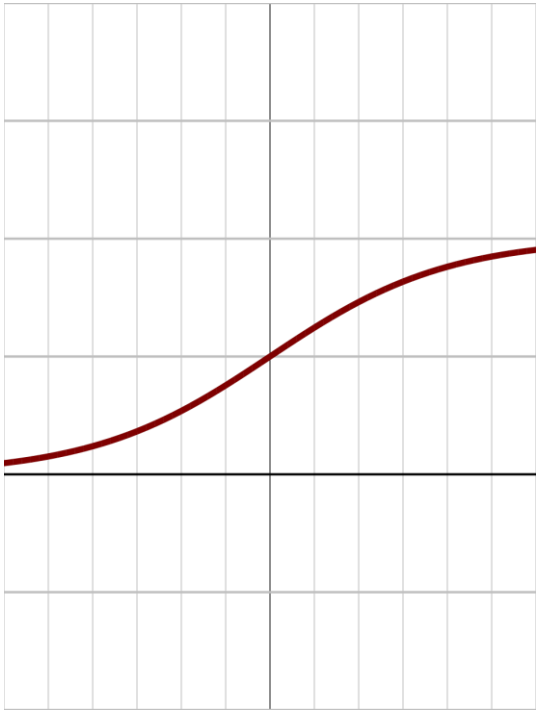
…

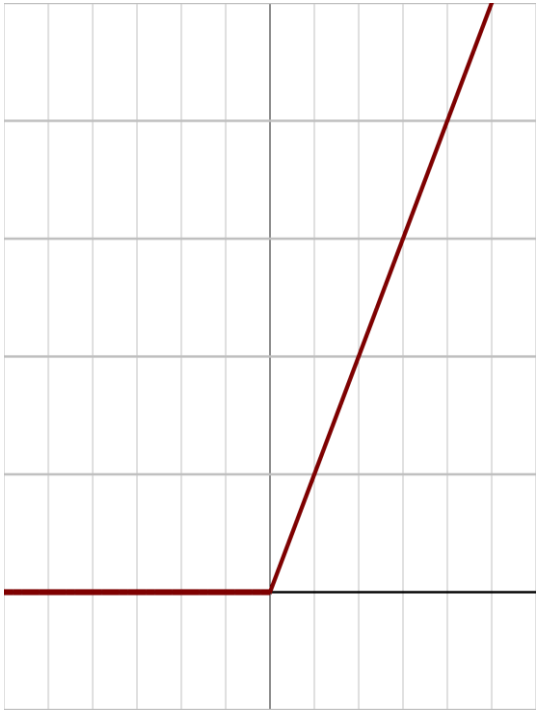
$$h_m(x) = \frac{1}{1 + e^{-g_m(x)}}$$

Final output

$$f(x) = u_1 h_1(x) + \cdots + u_m h_m(x)$$

Activation functions





- Sigmoid saturates at both ends (slope → 0)
- Rectified linear unit produces no response for x < 0
  - Default choice nowadays

Universal approximation (Cybenko, 1989)

Universal Approximation Theorem: Fix a continuous function $\sigma : \mathbb{R} \to \mathbb{R}$ (activation function) and positive integers d, D. The function σ is not a polynomial if and only if, for every continuous function $f : \mathbb{R}^d \to \mathbb{R}^D$ (target function), every compact subset K of $\mathbb{R}^d$, and every $\epsilon > 0$ there exists a continuous function $f_\epsilon : \mathbb{R}^d \to \mathbb{R}^D$ (the layer output) with representation

$$f_\epsilon = W_2 \circ \sigma \circ W_1,$$

where W_2, W_1 are composable affine maps and $\circ$ denotes component-wise composition, such that the approximation bound

$$\sup_{x \in K} \|f(x) - f_\epsilon(x)\| < \varepsilon$$

holds for any ϵ arbitrarily small (distance from f to f_ϵ can be infinitely small).

- MLP with just one hidden layer and non-polynomial activation function can approximate any continuous function with arbitrarily high precision

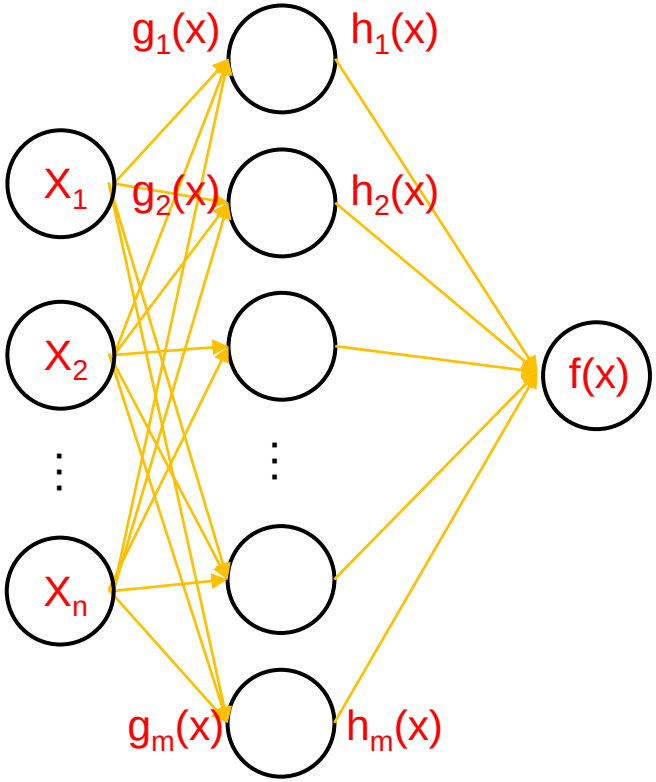
MLP learning algorithm

Gradient calculation with chain rule

- A one-layer setup:
  - $g_i(x) = w_{i,1}x_1 + \cdots + w_{i,n}x_n$
  - Sigmoid activation: $h_i(x) = \frac{1}{1+e^{-g_i(x)}}$
  - Final output: $f(x) = u_1 h_1(x) + \cdots + u_m h_m(x)$
  - MSE loss: $L(f(x), y) = \frac{1}{2}\|f(x) - y\|^2$

- Gradients through chain rule:
  - $\frac{\delta L}{\delta u_i} = \frac{\delta L}{\delta f}\frac{\delta f}{\delta u_i} = (f(x) - y) \cdot h_i(x)$
  - $\frac{\delta h_i}{\delta w_{i,j}} = \frac{\delta h_i}{\delta g_i}\frac{\delta g_i}{\delta w_{i,j}} = g_i(\mathrm{x})(1 - g_i(\mathrm{x}))\, x_j$
  - $\frac{\delta L}{\delta w_{i,j}} = \frac{\delta L}{\delta f}\frac{\delta f}{\delta h_i}\frac{\delta h_i}{\delta w_{i,j}} = (f(x) - y) \cdot u_i \cdot g_i(\mathrm{x})(1 - g_i(\mathrm{x}))\, x_j$

Backpropagation



- Work backward from output layer to input layer
- Memorize all parameters and input/output values into every neuron
- Compute gradient for each parameter using the chain rule
- The number of multiplicative terms in the chain rule scales with the number of layers
  - Vanishing gradient problem when |gradient| < 1

Stochastic gradient descent

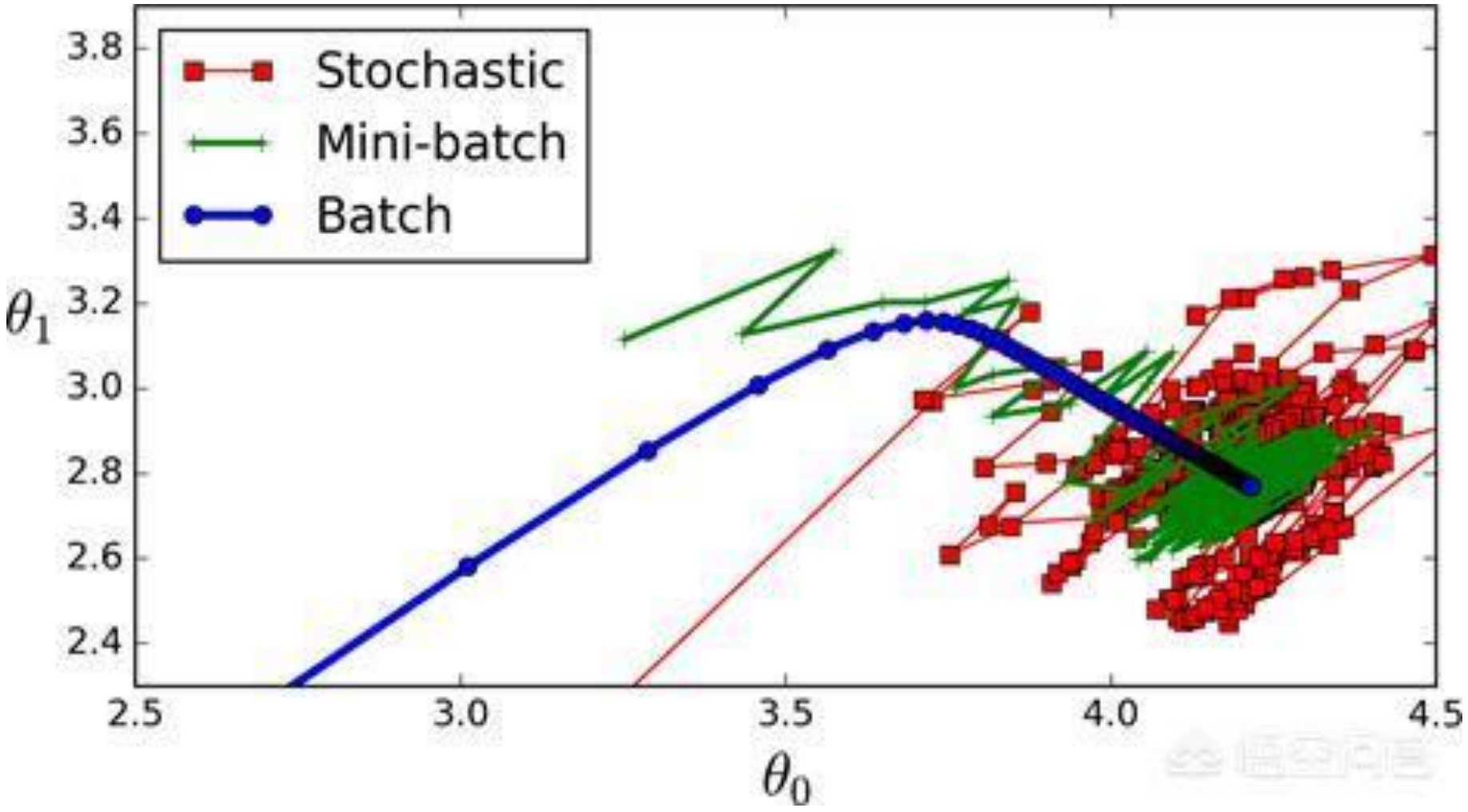
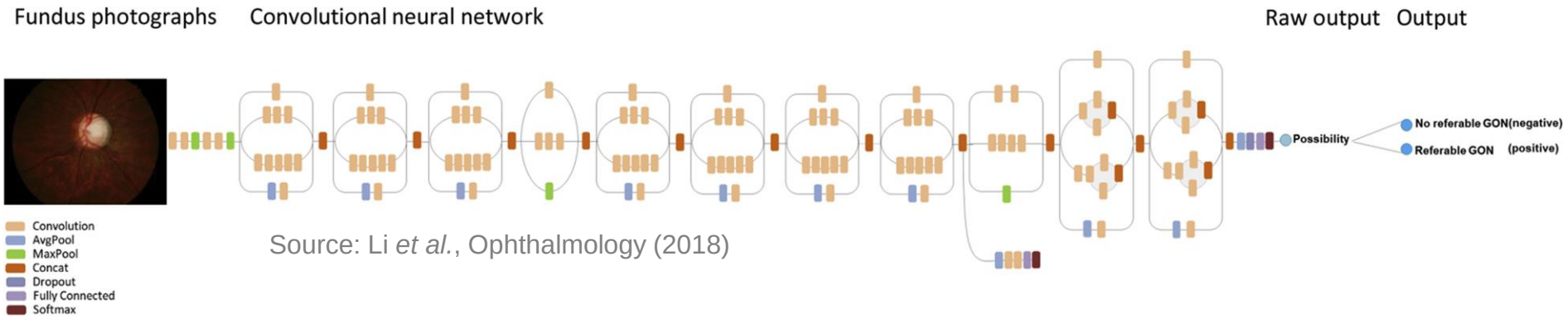


Image from programmersought.com

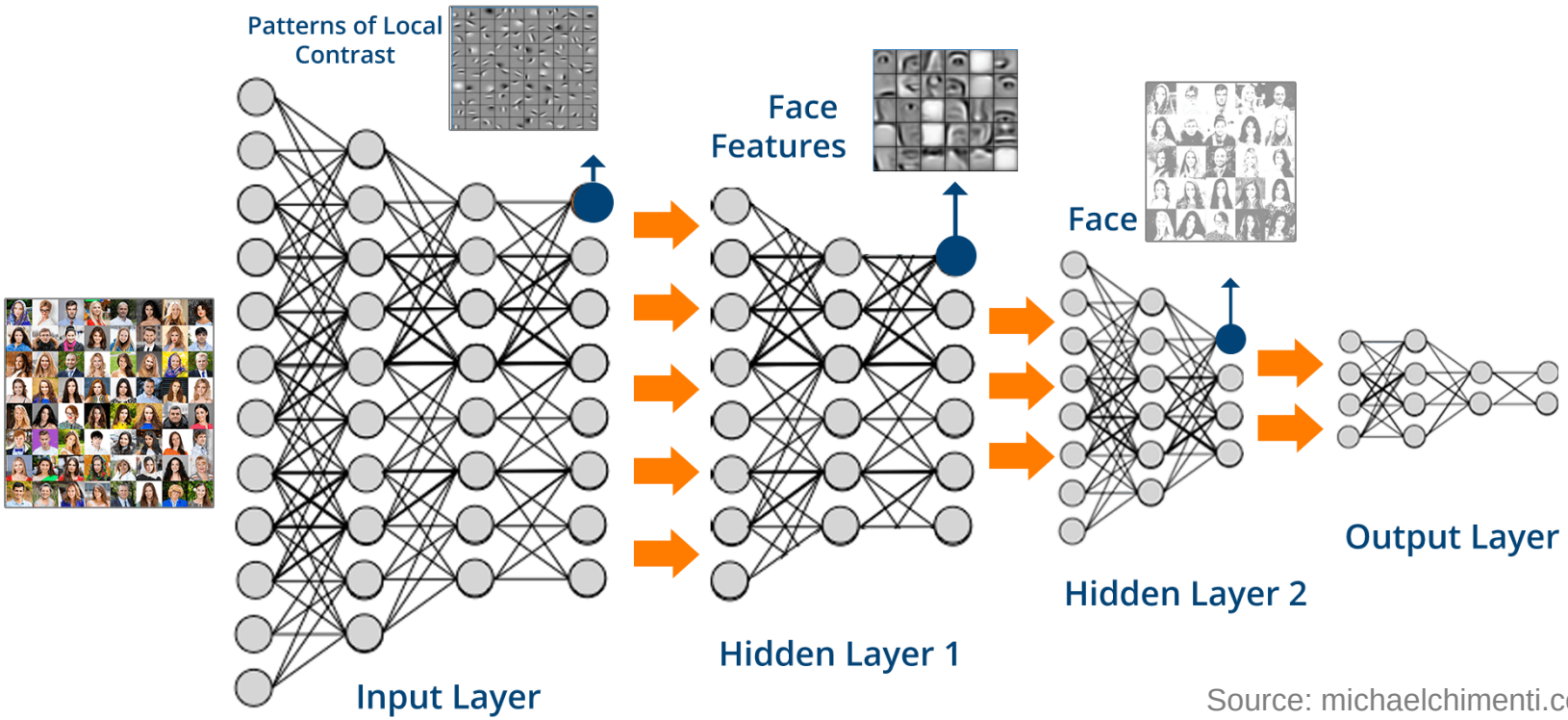
Deep learning

Deep learning in a nutshell

Deep learning = deep ANN



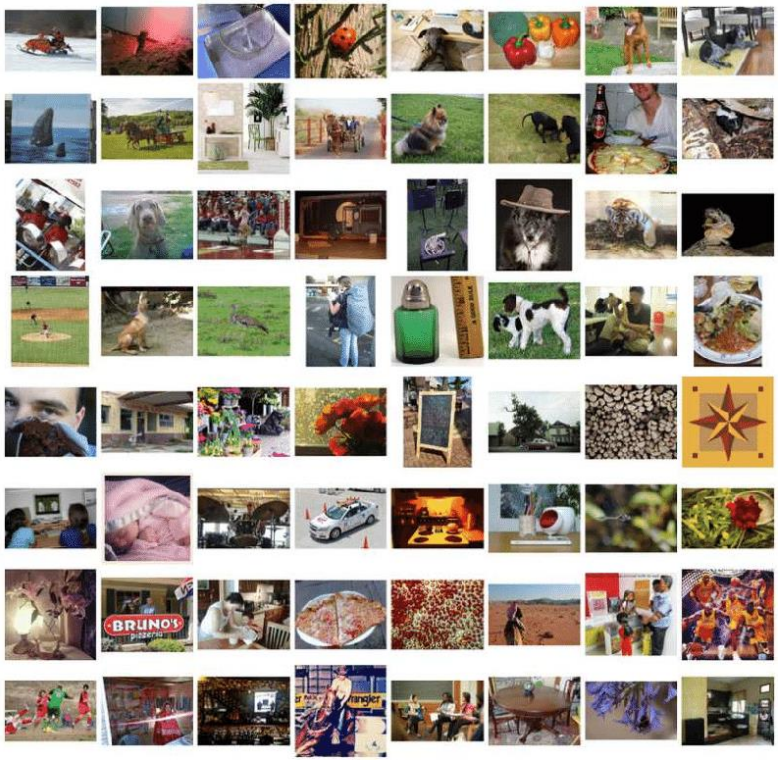
Source: Li *et al.*, Ophthalmology (2018)



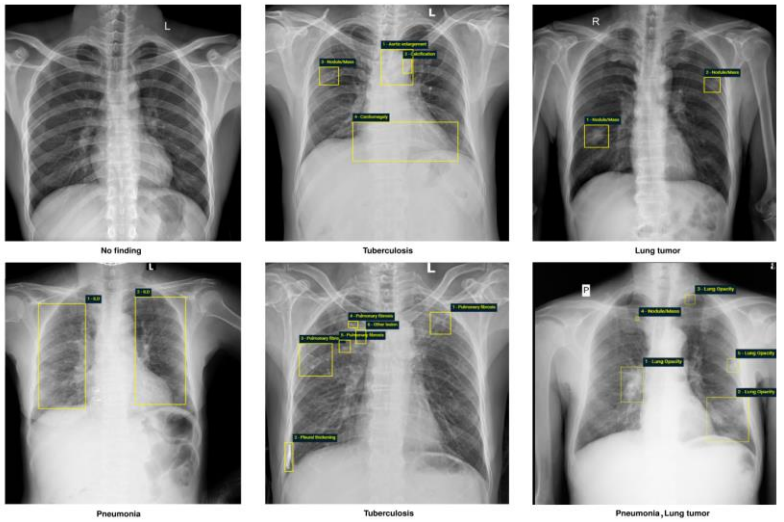
Source: michaelchimenti.com

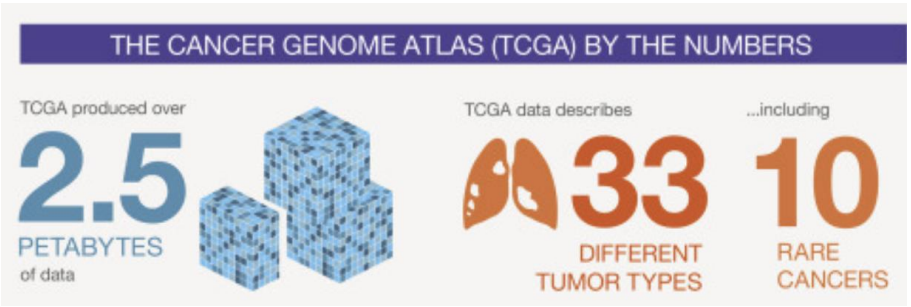
Ingredient #1: Annotated digital data

ImageNet



CXR datasets





- ImageNet is a collection of millions of annotated general images that kickstart modern deep learning

Ingredient #2: Computing hardware

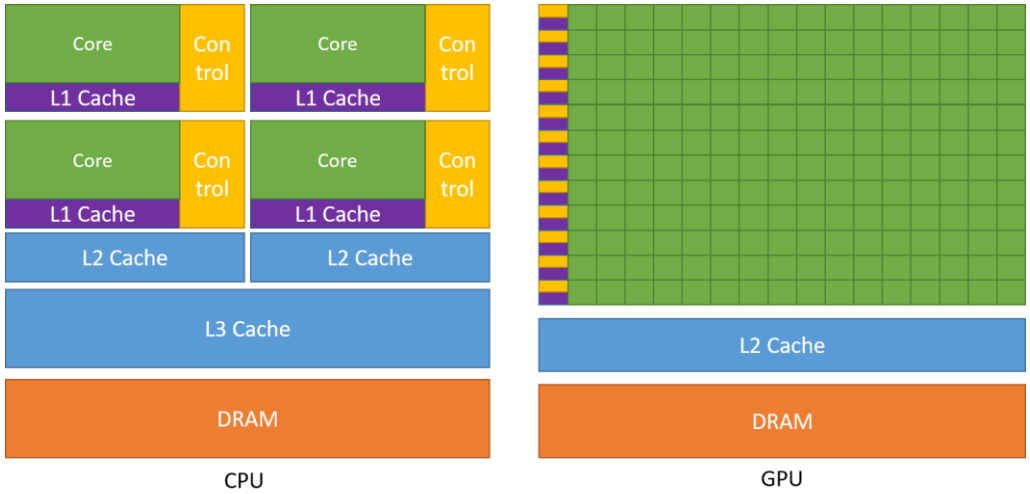
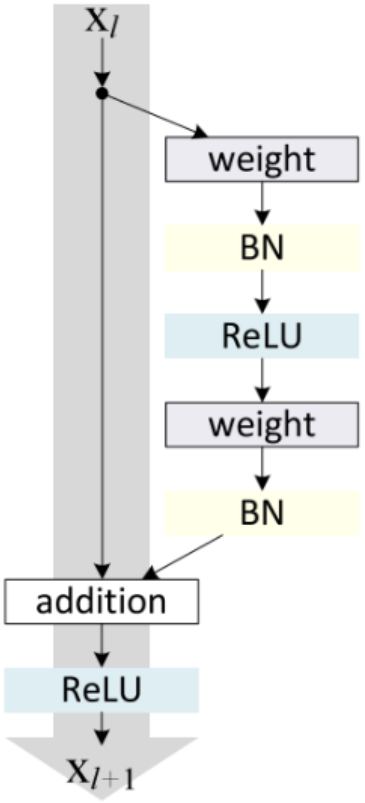


Image from analyticsvidhya.com

- CPU = few high capability compute unit
- GPU = swarm of low capability compute unit

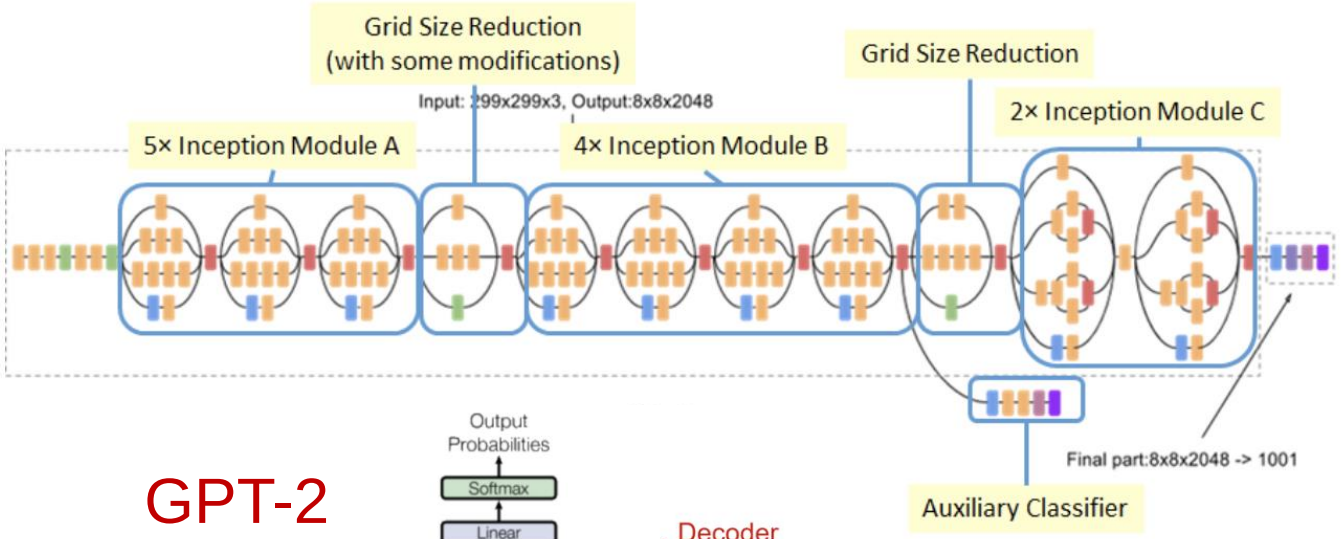
Ingredient #3: ANN architectures

ResNet

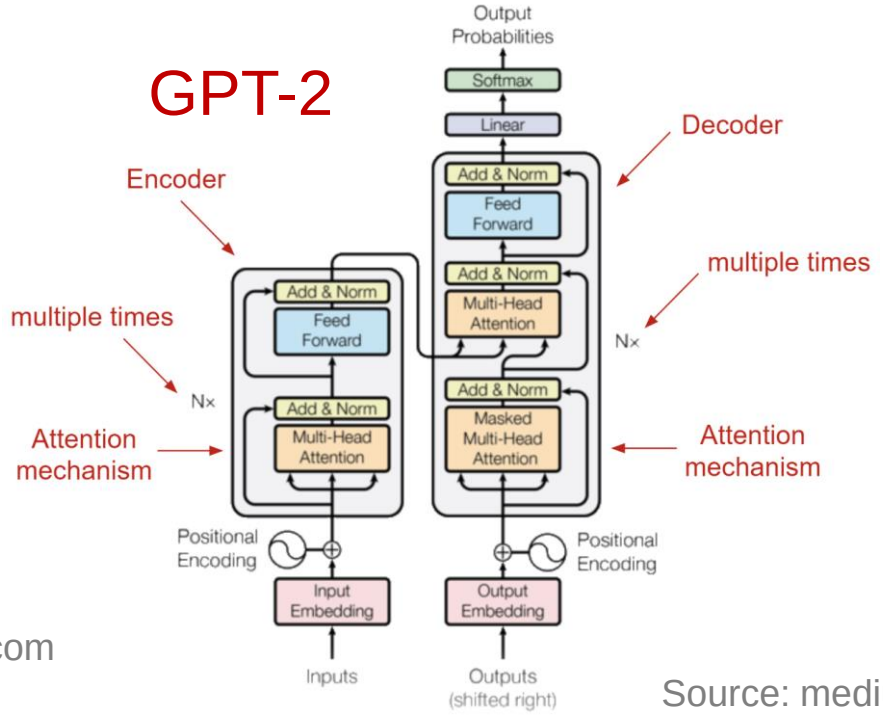


Source: towarddatascience.com

Inception

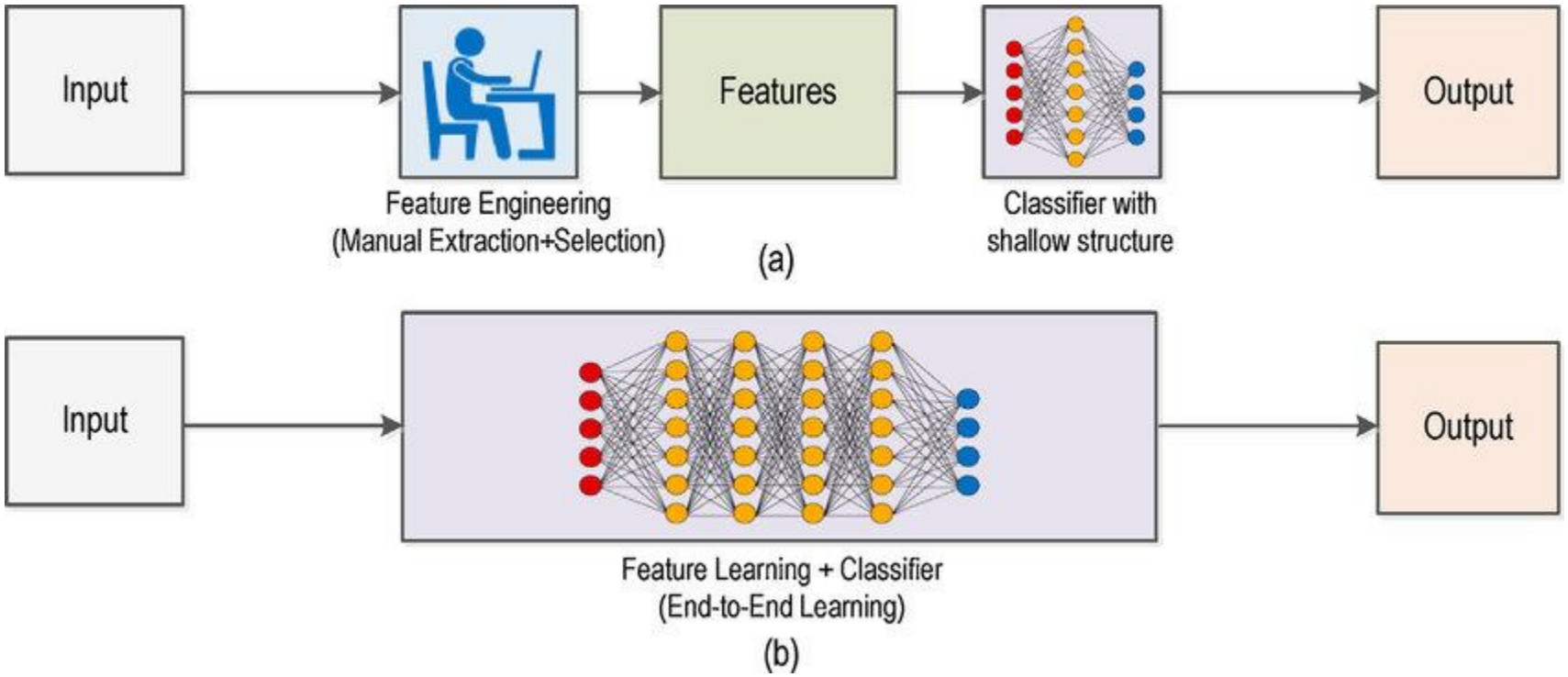


GPT-2



Source: medium.com

End-to-end capability



Source: Wang *et al.*, J Manufacturing Systems (2018)

- Deep neural network can extract informative features from raw data such as images or texts with minimal human help
  - Representation learning

Representation learning view of deep learning

Representation learning

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Image from hackermoon.com

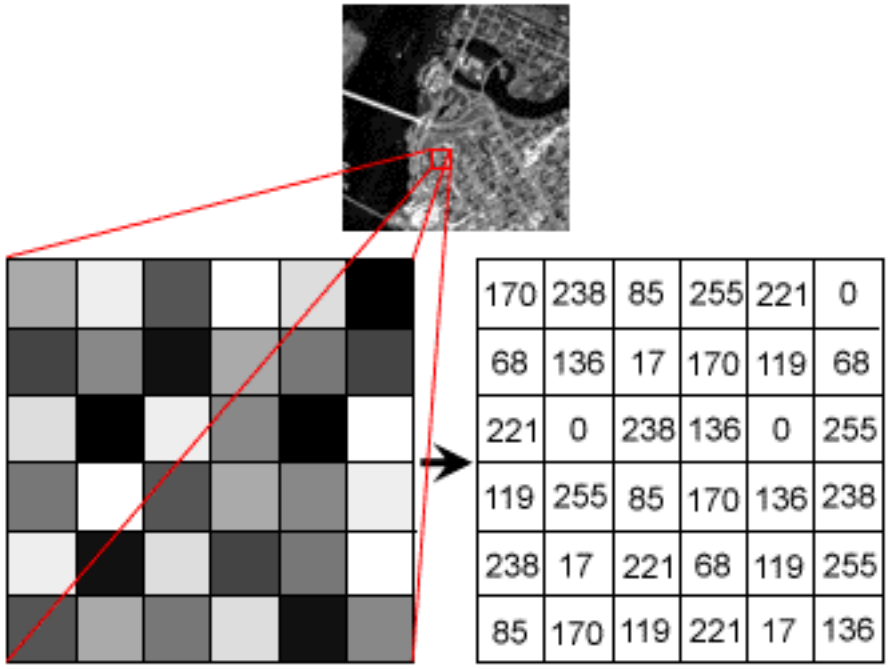
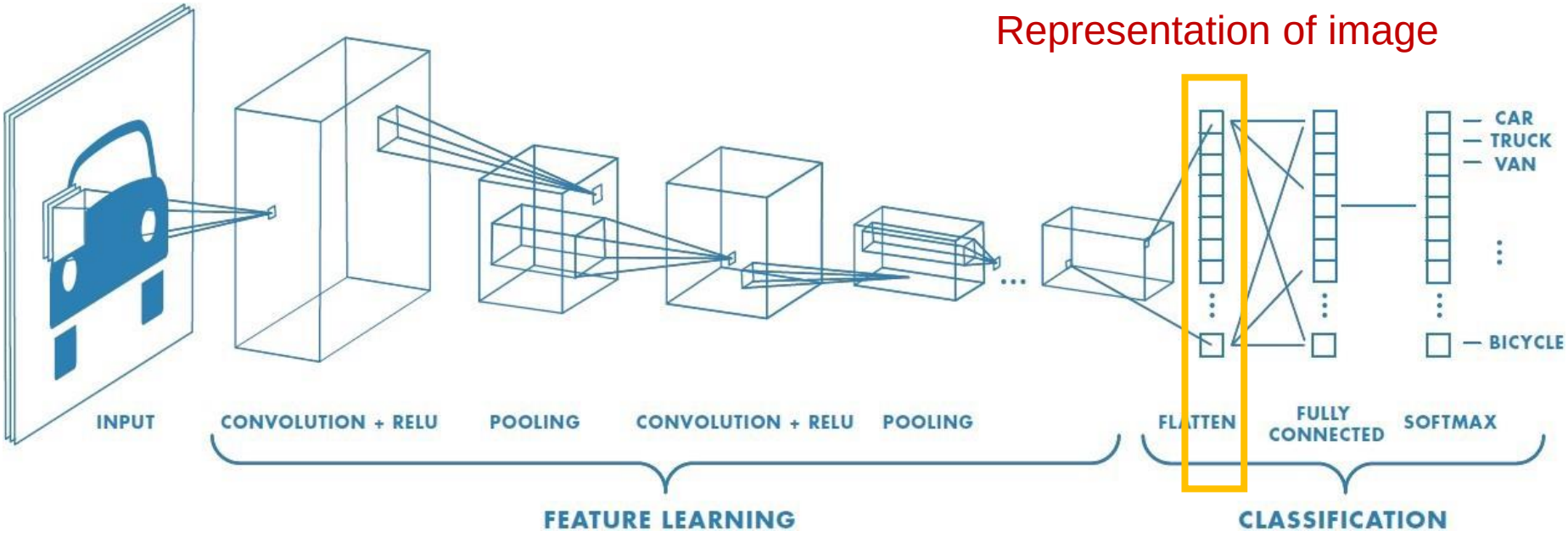


Image from naushardsblog.wordpress.com

- Representation = presentation of information contained in the data
- Simple representation is not useful for learning

Image representation



Source: towardsdatascience.com by Saha, S.

- Representation learning = machine learning for obtaining a good representation of the data
- Example: convolution layers of a CNN

Word representation

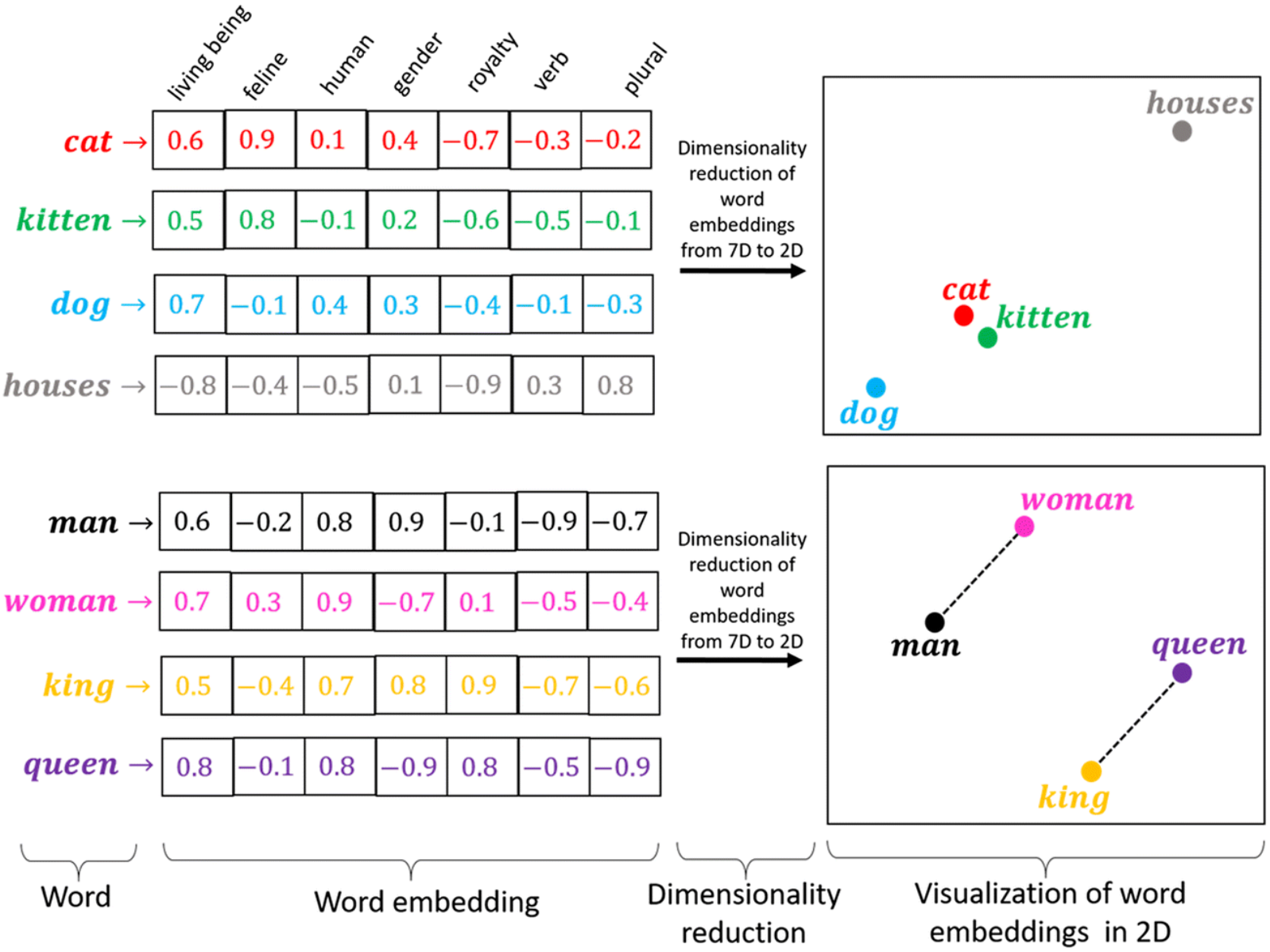
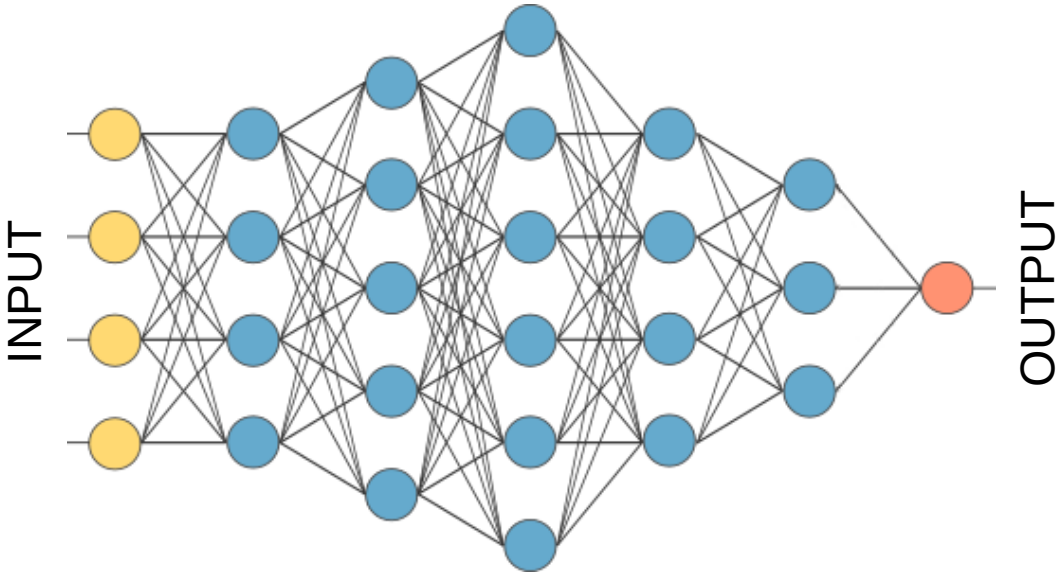


Image from medium.com

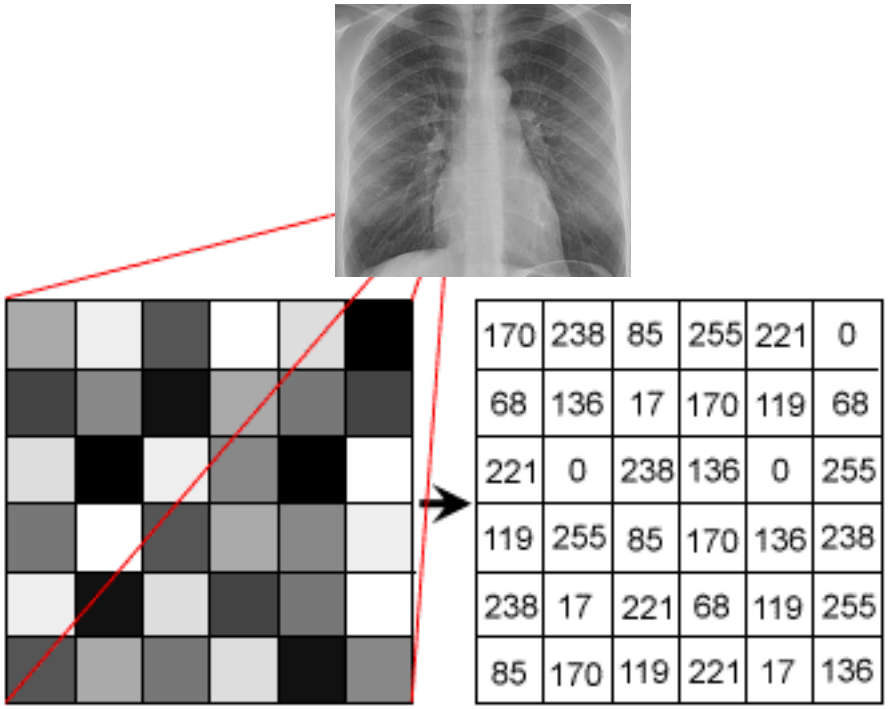
The need for new ANN architectures

Limitation of fully connected network



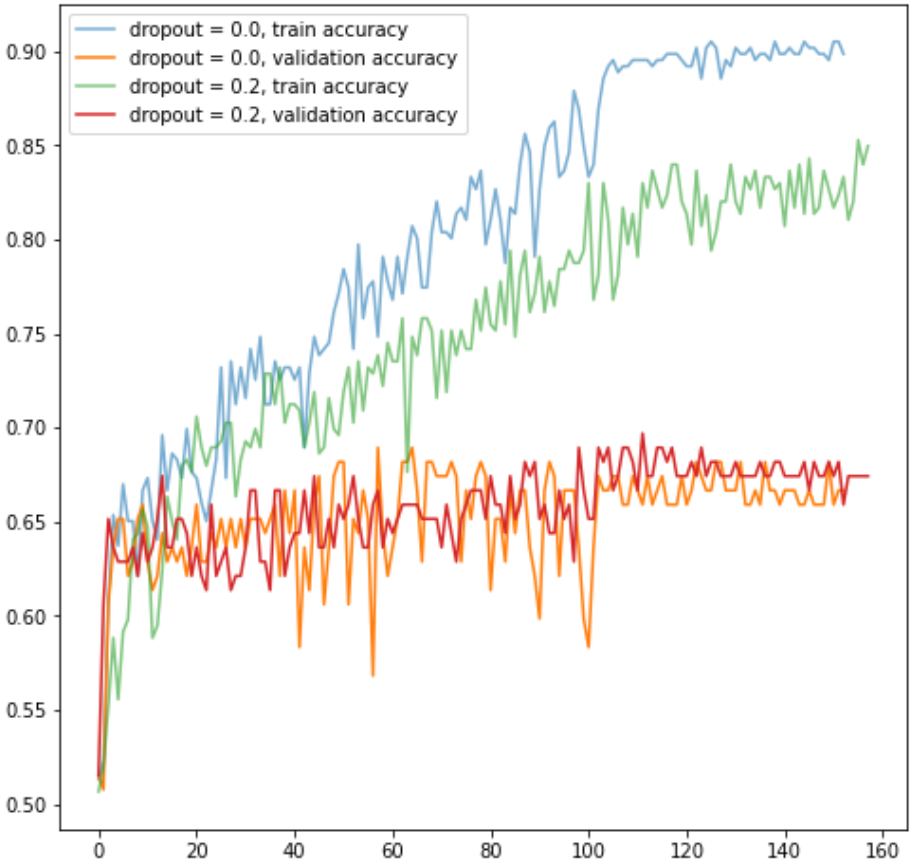
- What if features have contextual relationships?
  - Adjacent pixels in an image
  - Adjacent words in a sentence
- What if input size is variable?
  - Number of words in a medical report

Fully connected network's image perception



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Image from naushardsblog.wordpress.com

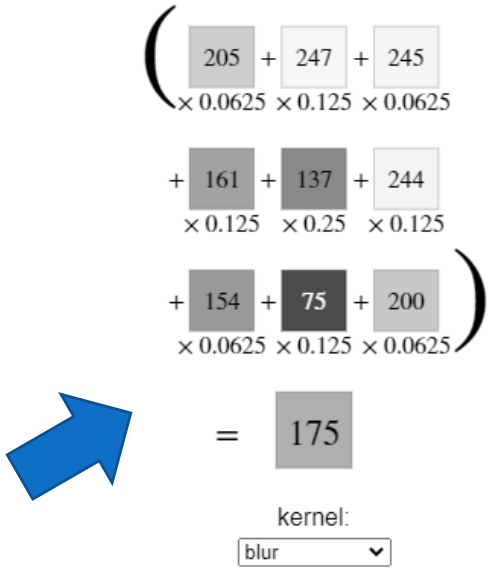


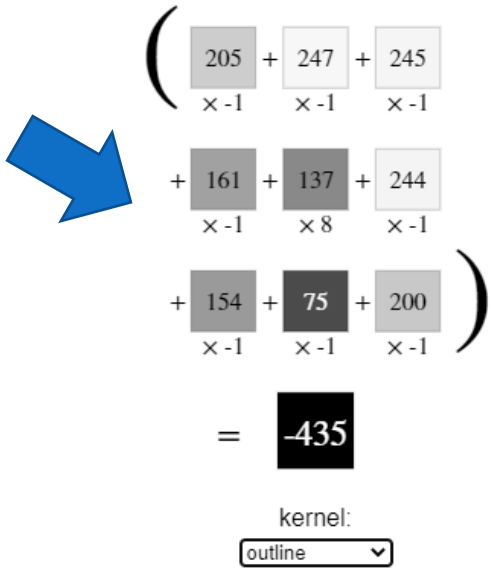
- Fully connected network cannot learn anything from pixel values of chest x-ray images

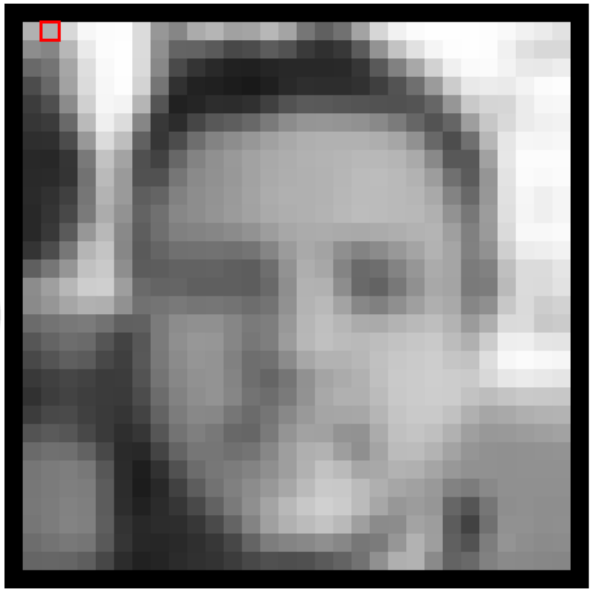
Feature extraction with filters

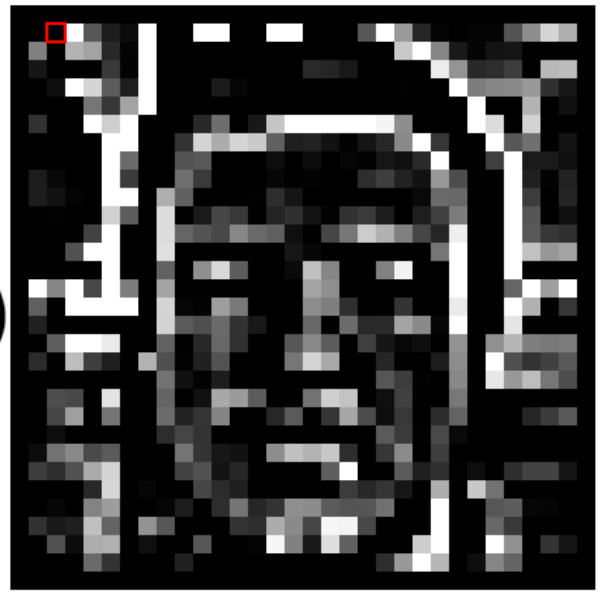
input image

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



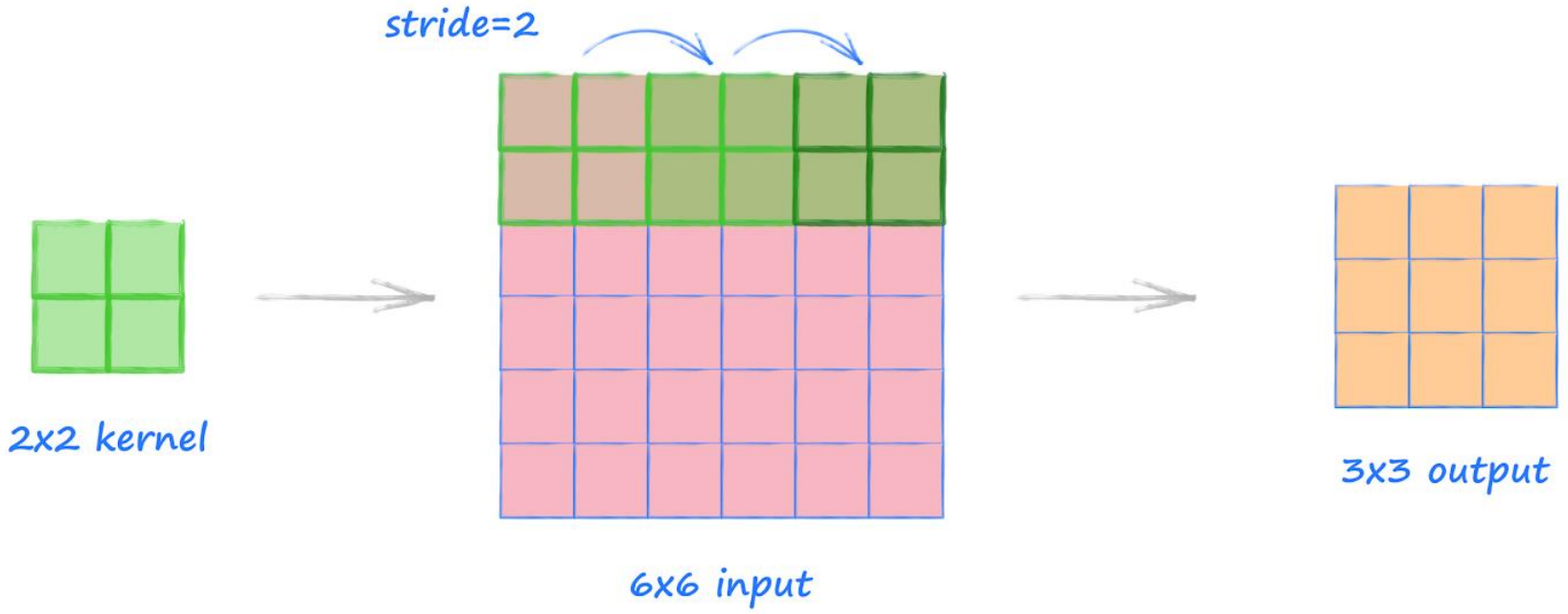






output image

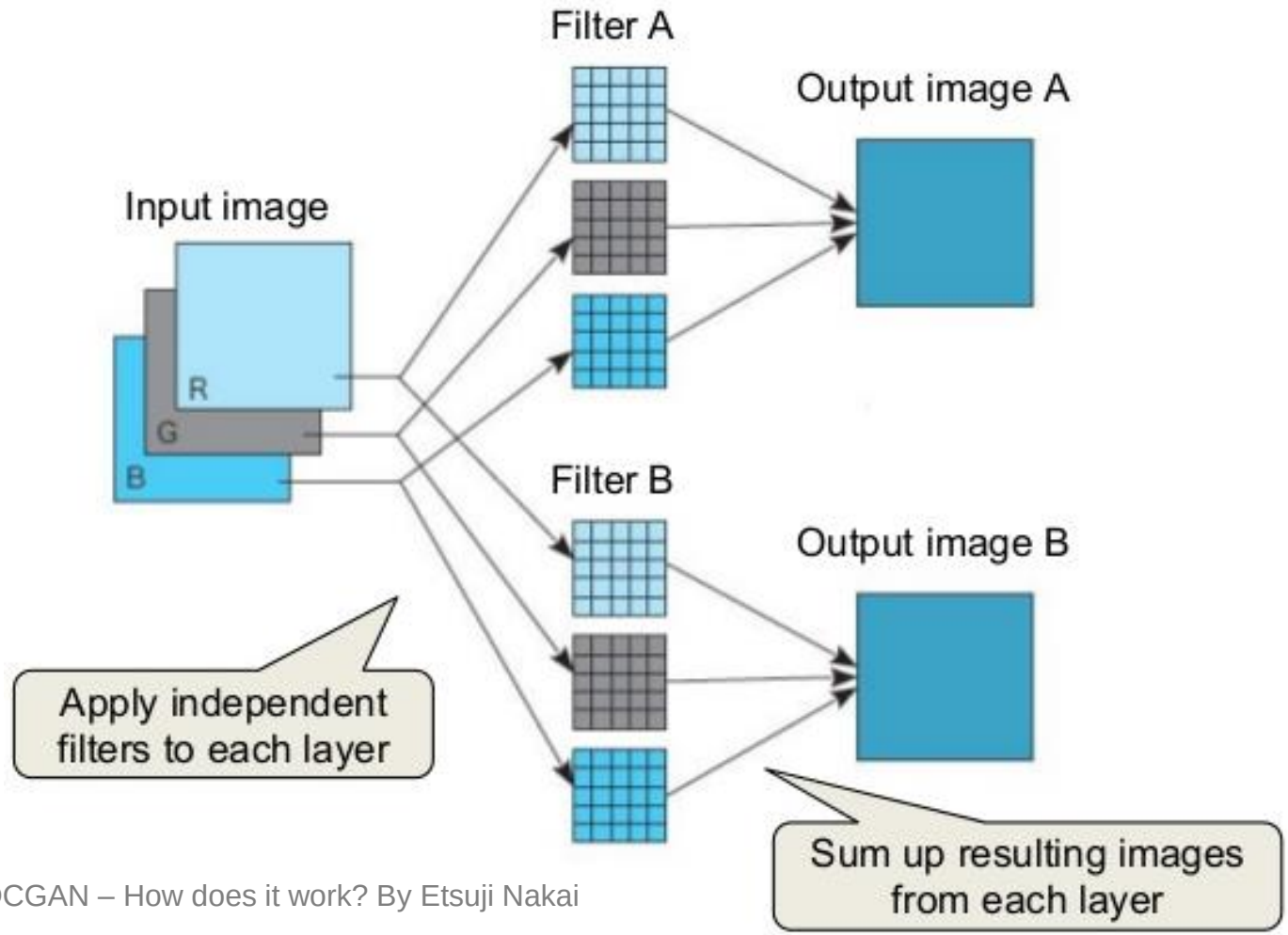
Convolutional operation



Source: makeyourownneuralnetwork.blogspot.com

- Larger stride further reduces the output size in exchange for some loss in resolution

Image feature extraction

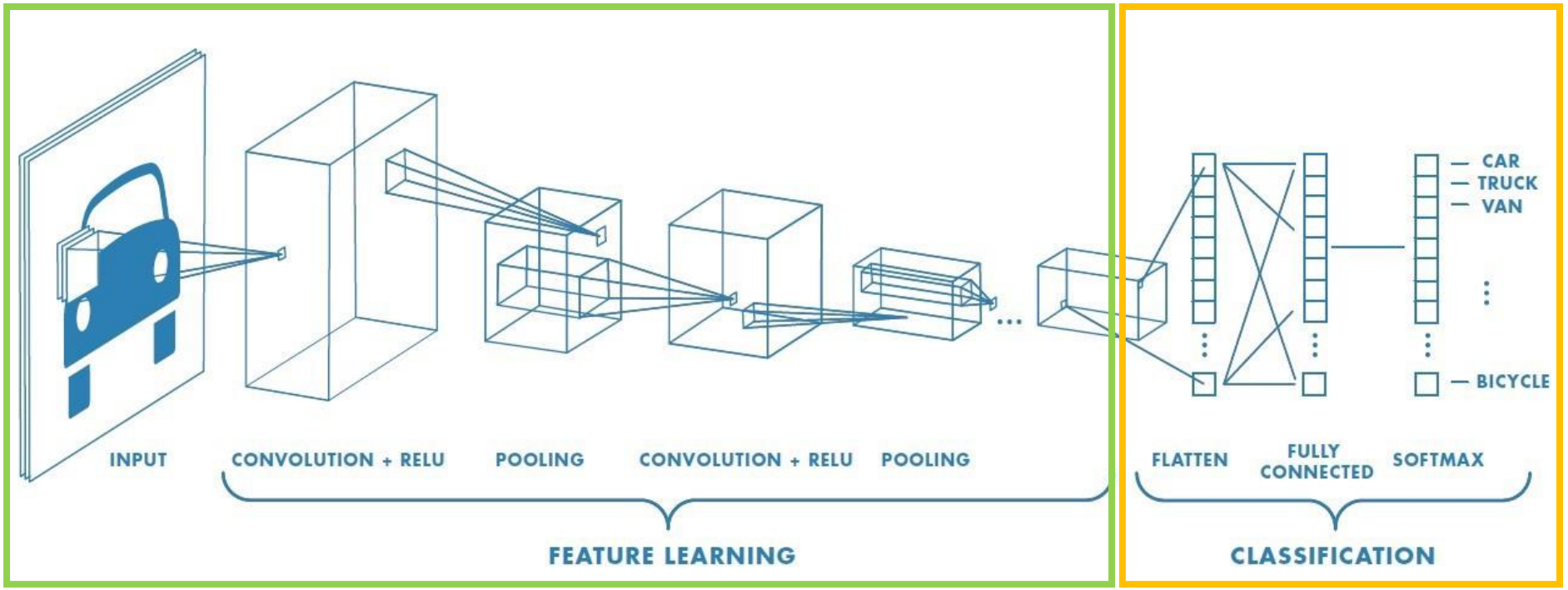


Source: DCGAN – How does it work? By Etsuji Nakai

- More filters generates more features

Convolutional neural network

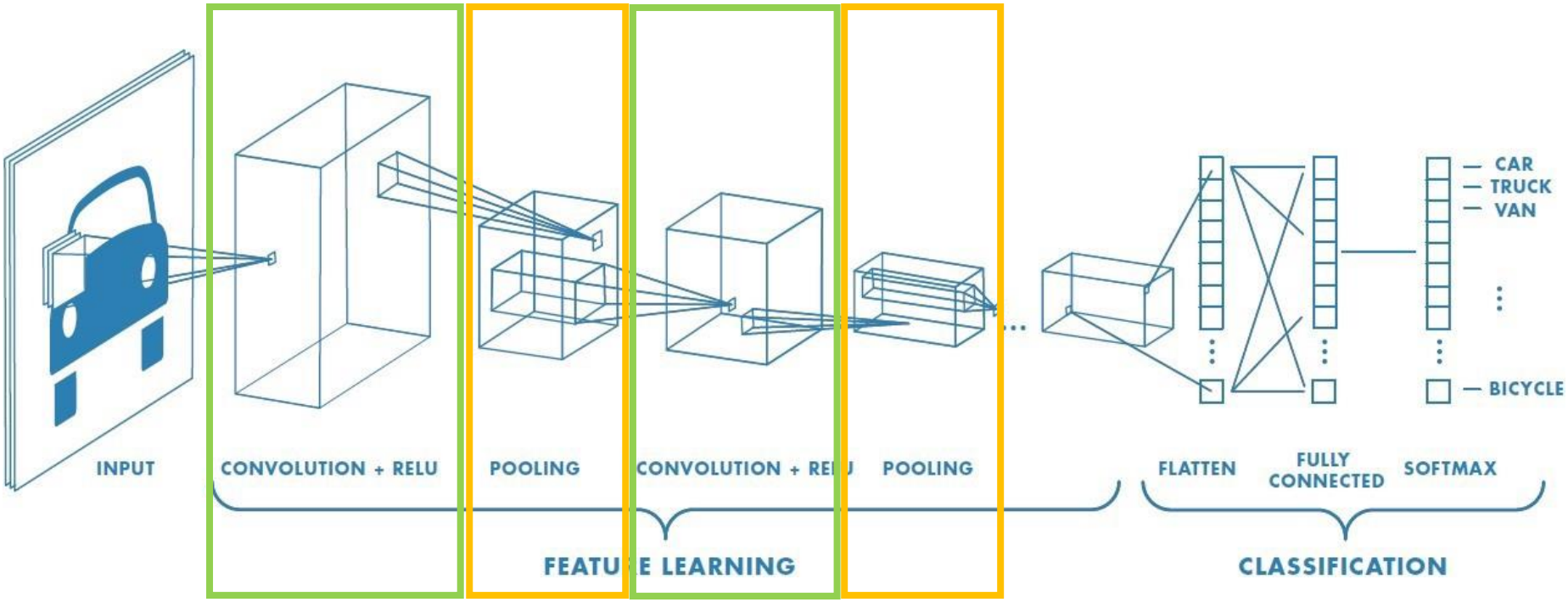
Convolutional neural network



Source: towardsdatascience.com by Saha, S.

- Data-driven, automatic search for optimal filters
  - Similar objective as radiomics, but no hand-crafting
- CNN layers extract features + fully-connect layers performing prediction

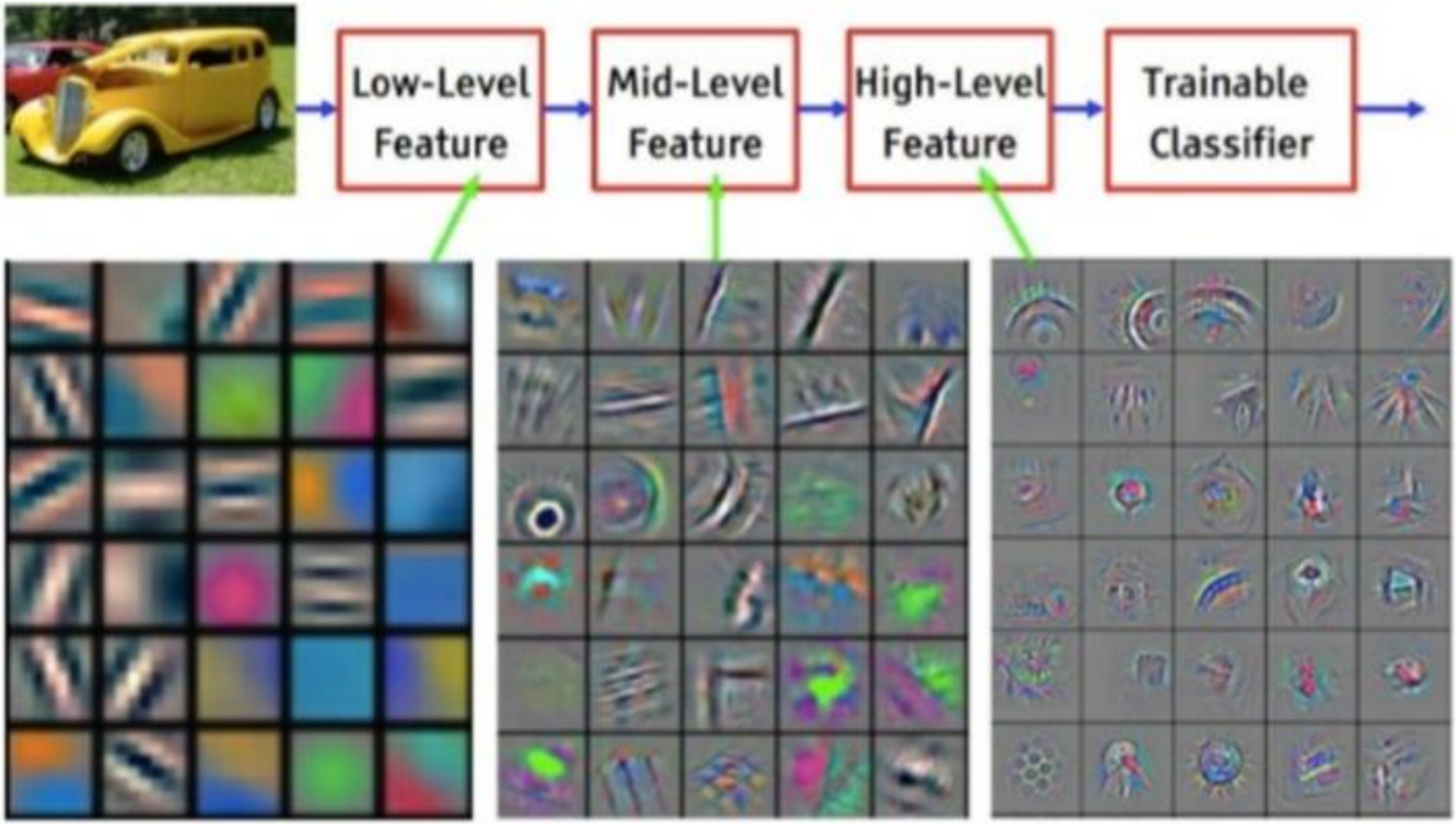
CNN block architecture



Source: towardsdatascience.com by Saha, S.

- Convolutional layer = compute feature
- Activation layer = apply non-linear transformation
- Pooling layer = reduce dimension

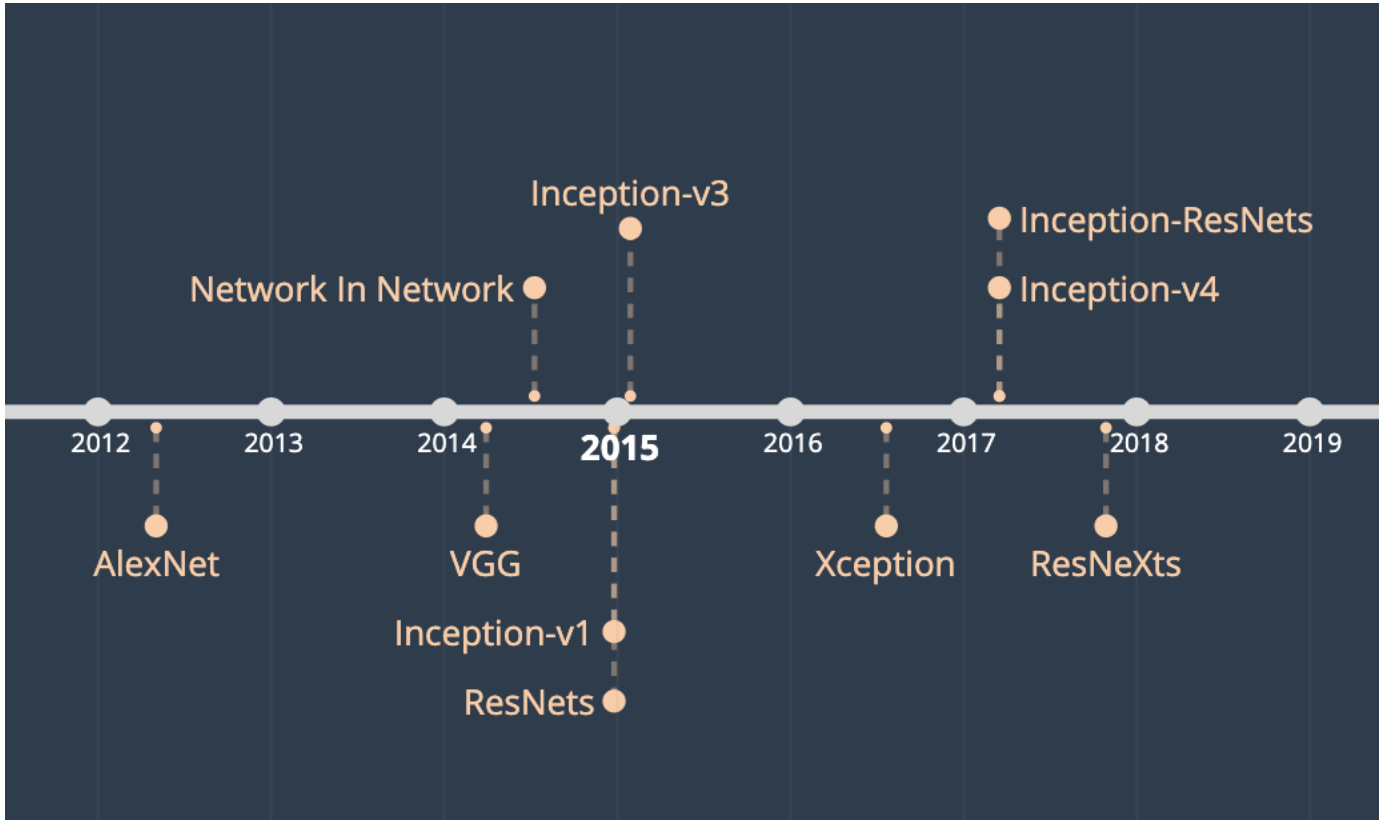
What happens inside a deep CNN?



Source: Zeiler and Fergus (2013)

- Early layers produce simple geometric features
- Deeper layers produce complex shapes and patterns

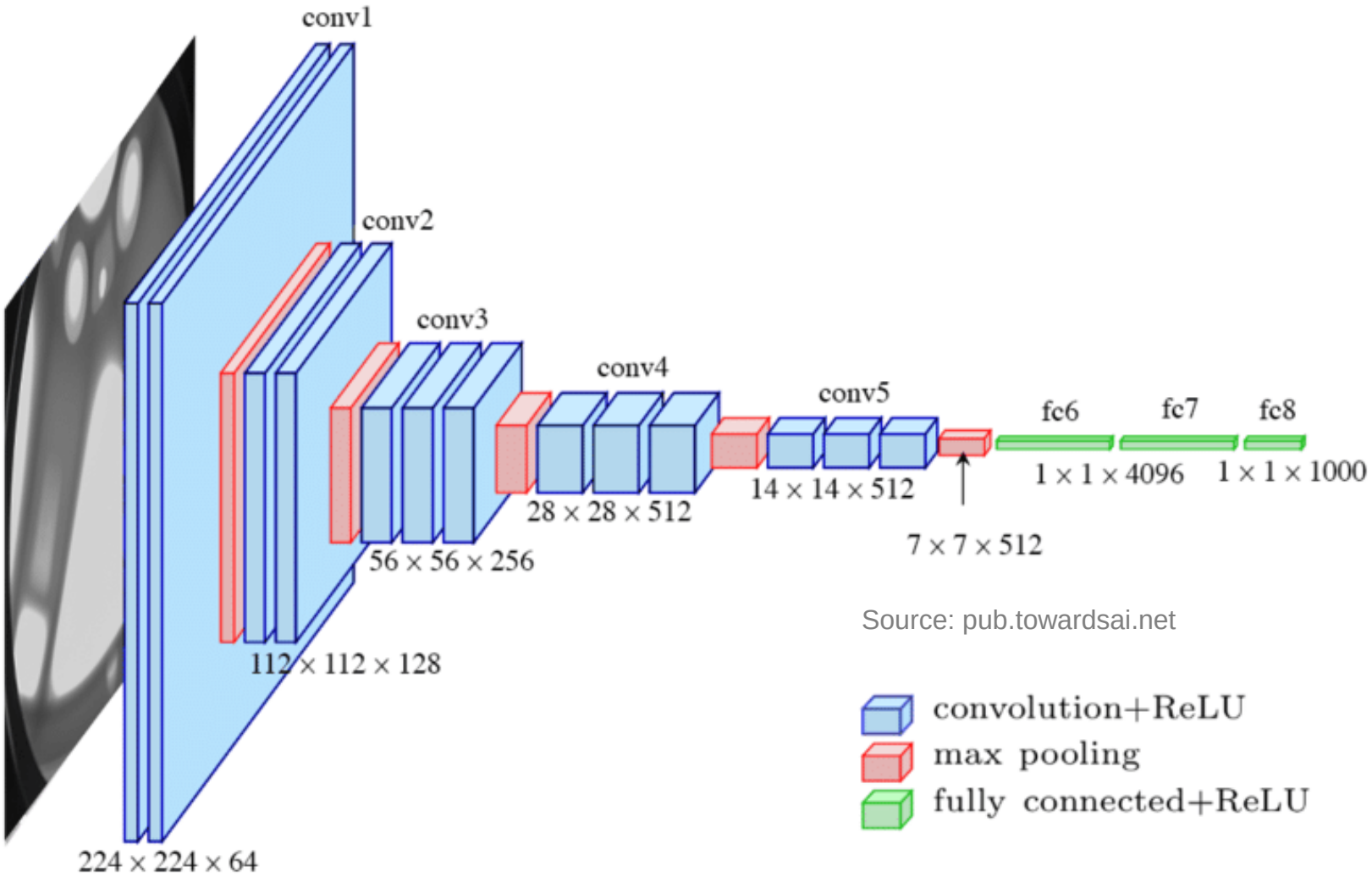
Timeline of popular CNN architectures



Source: towardsdatascience.com

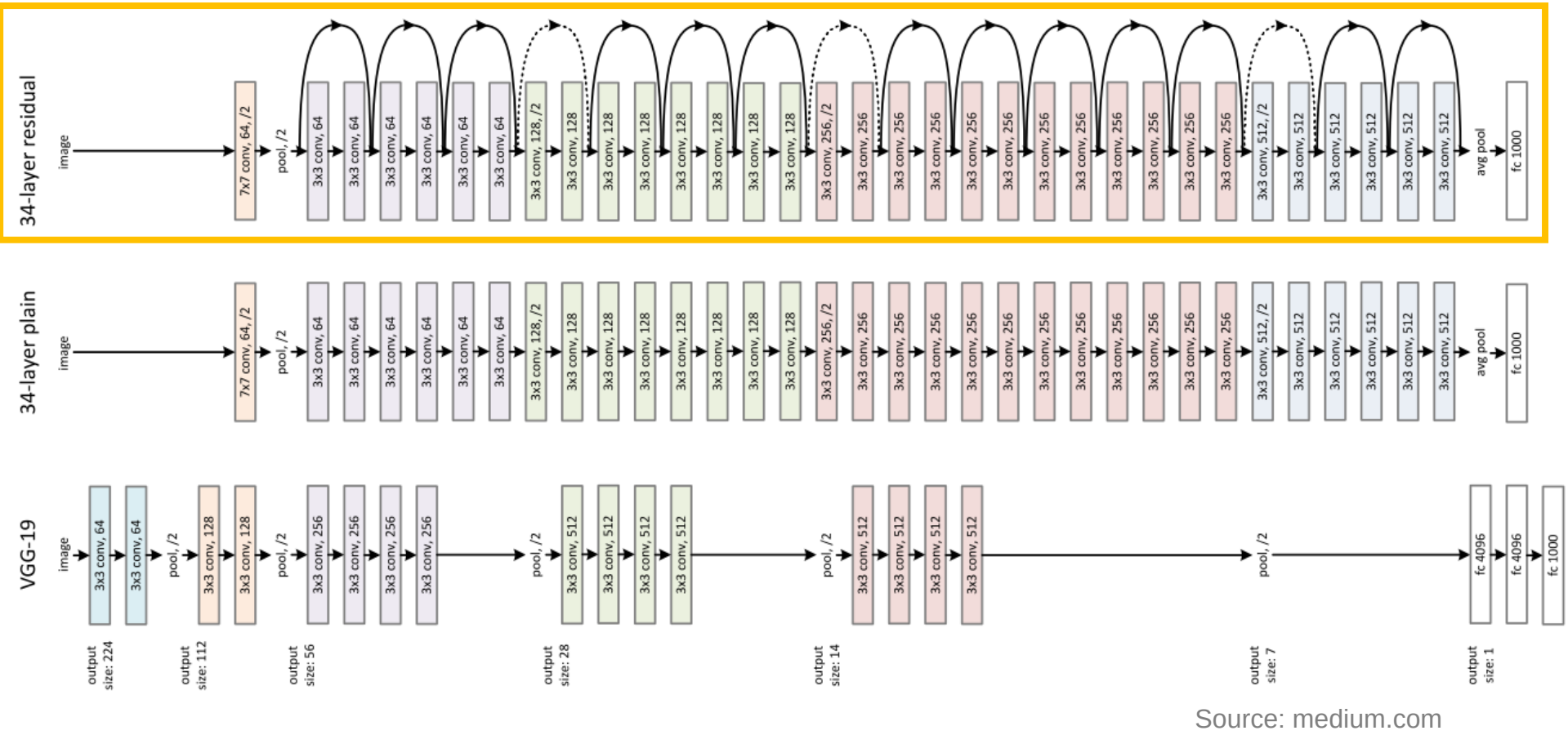
- AlexNet was the winner of 2012 ImageNet challenge
  - 5 convolution + 3 fully connected layers
  - Beat 2nd place by more than 10 percentage points

VGG-16



- Decrease image size, increase number of channels

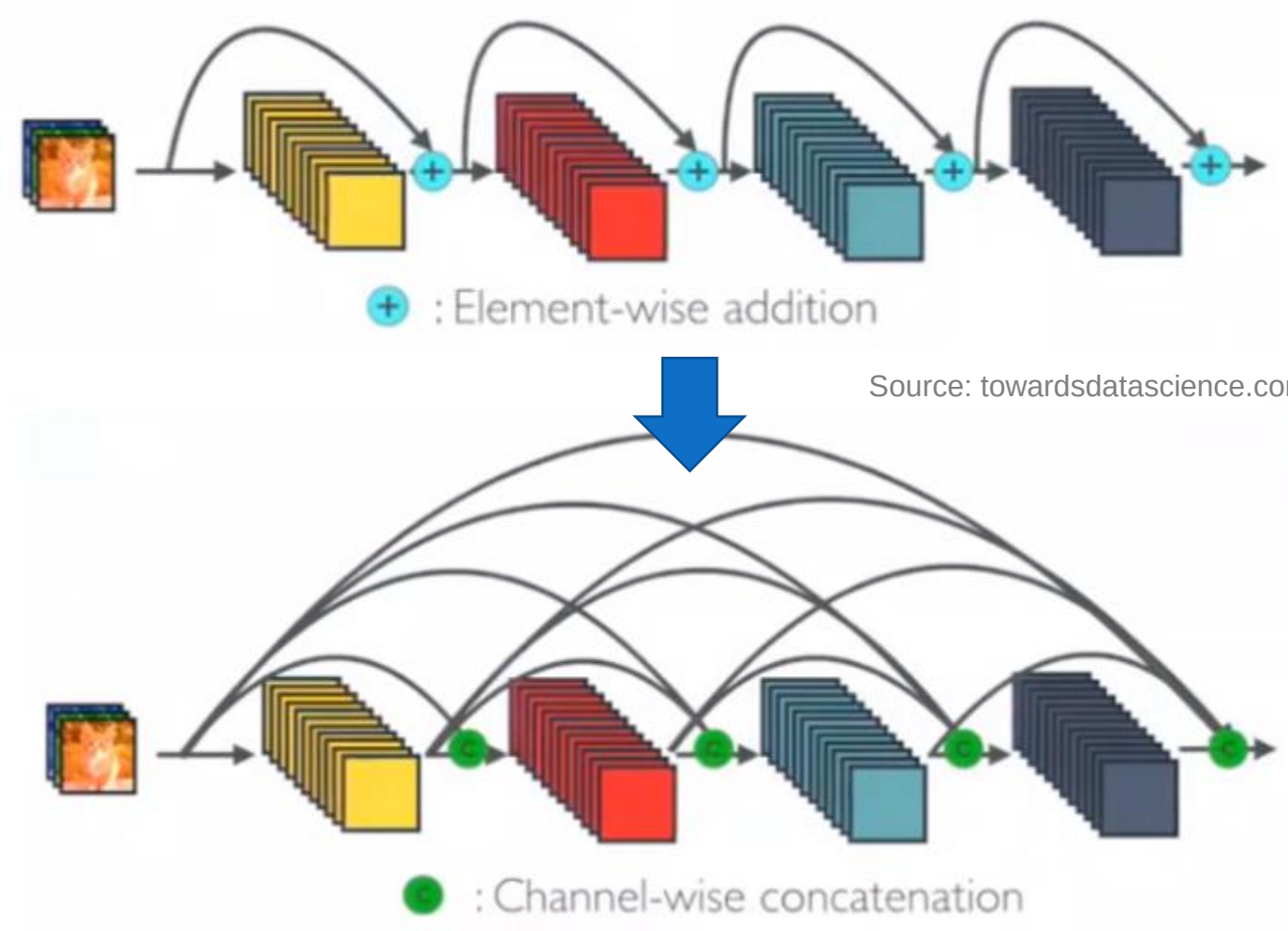
Residual Network (ResNet)



Source: medium.com

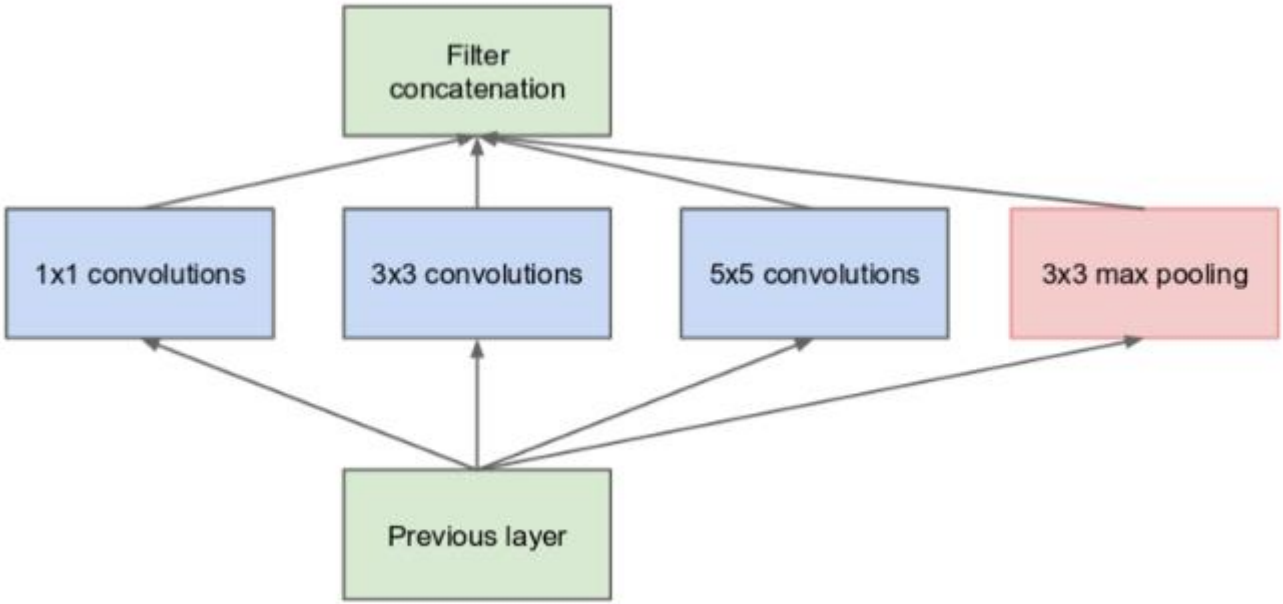
- Add bypasses over multiple convolutional layers
  - Reduce the number of multiplication terms in chain rule during backpropagation for early layers

DenseNet



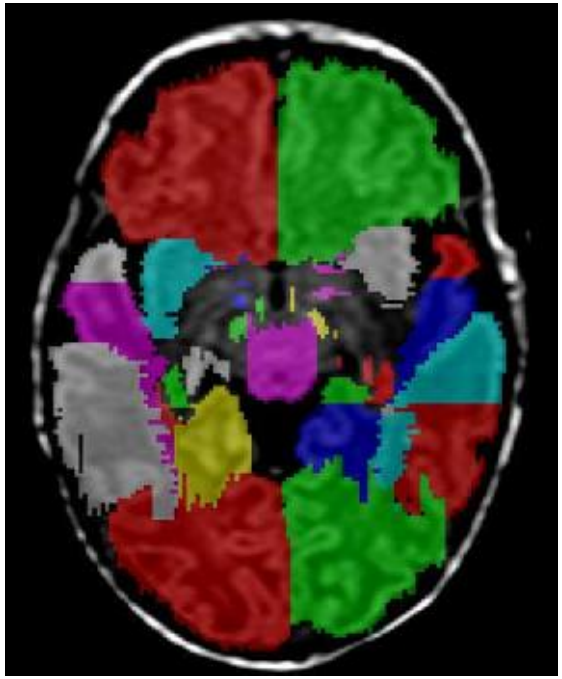
- Feedforward bypasses

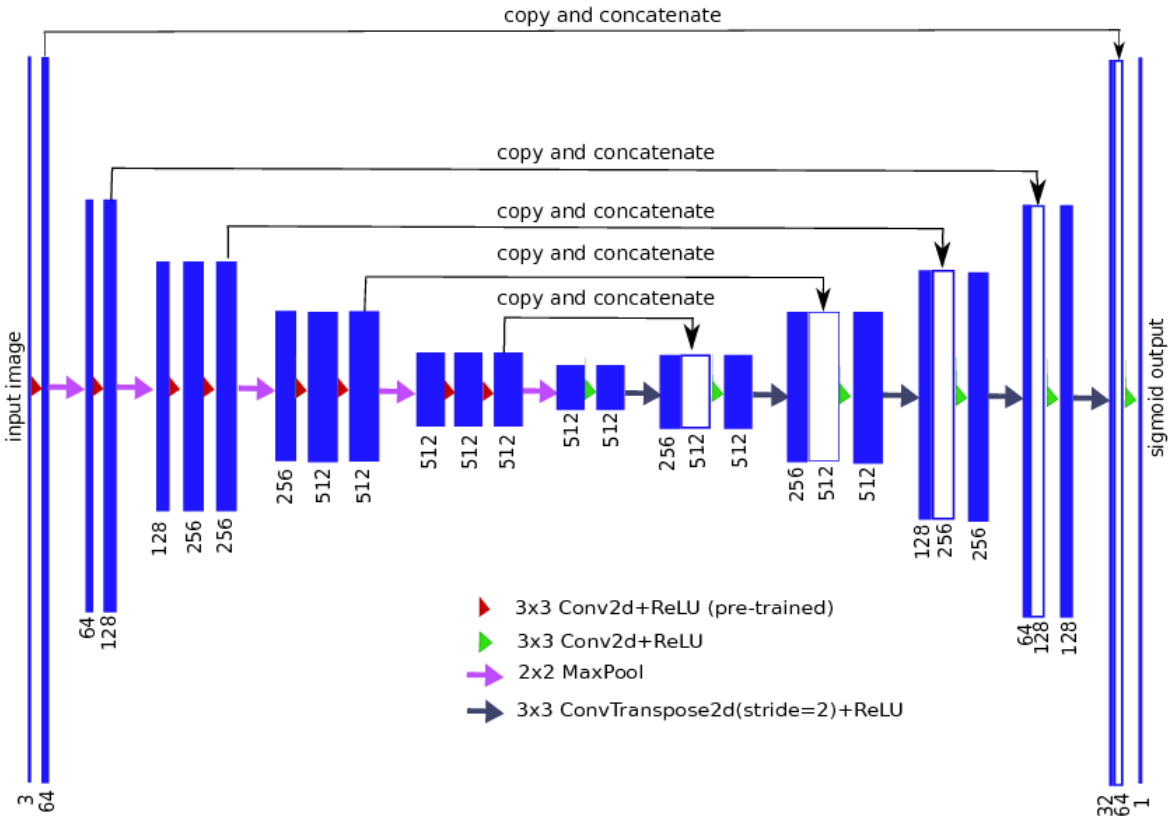
Inception = multi-resolution CNN



From left: A dog occupying most of the image, a dog occupying a part of it, and a dog occupying very little space (Images obtained from Unsplash).

Image segmentation = pixel classification



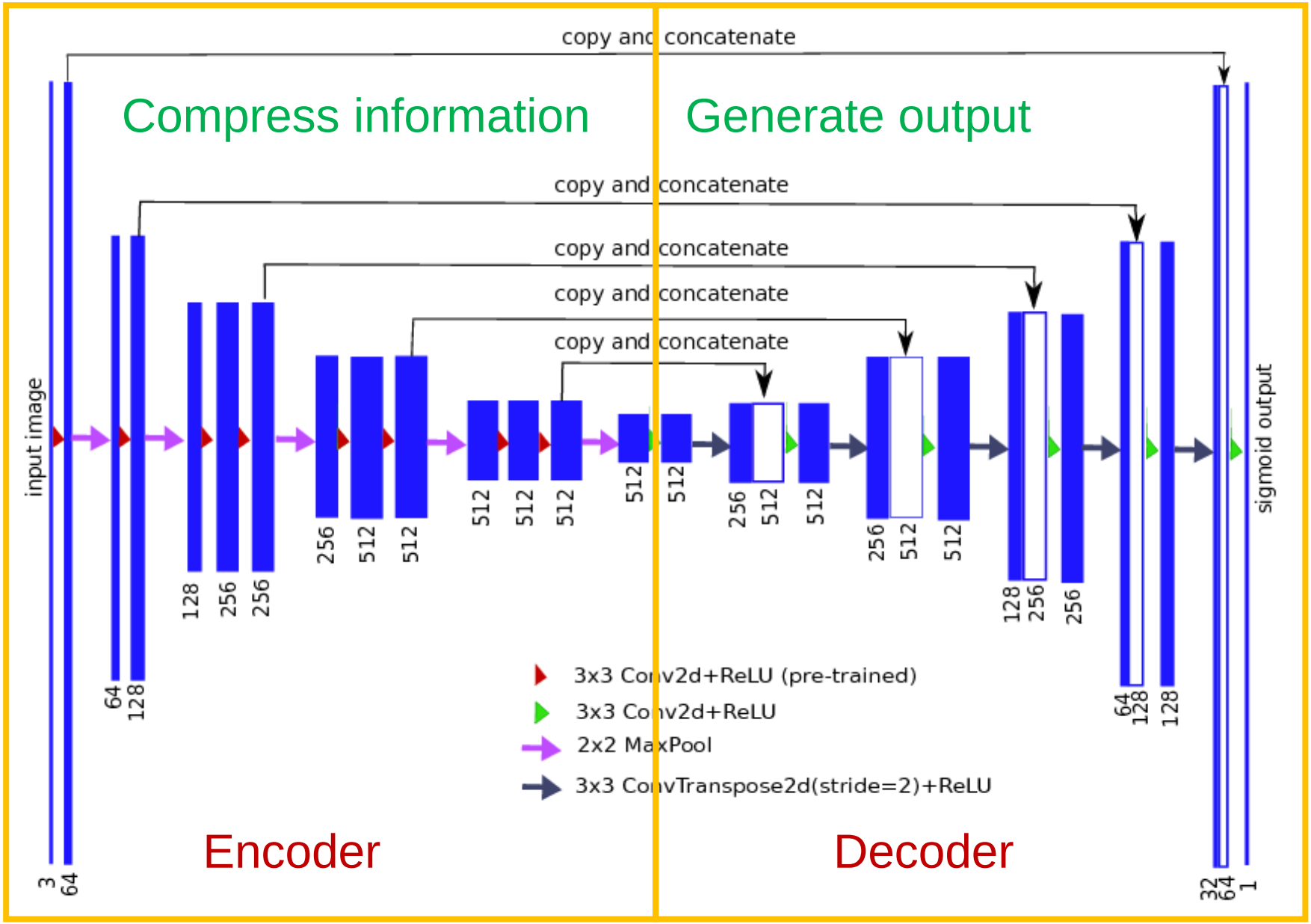


Makropoulos et al. IEEE Trans Med Imaging (2014)

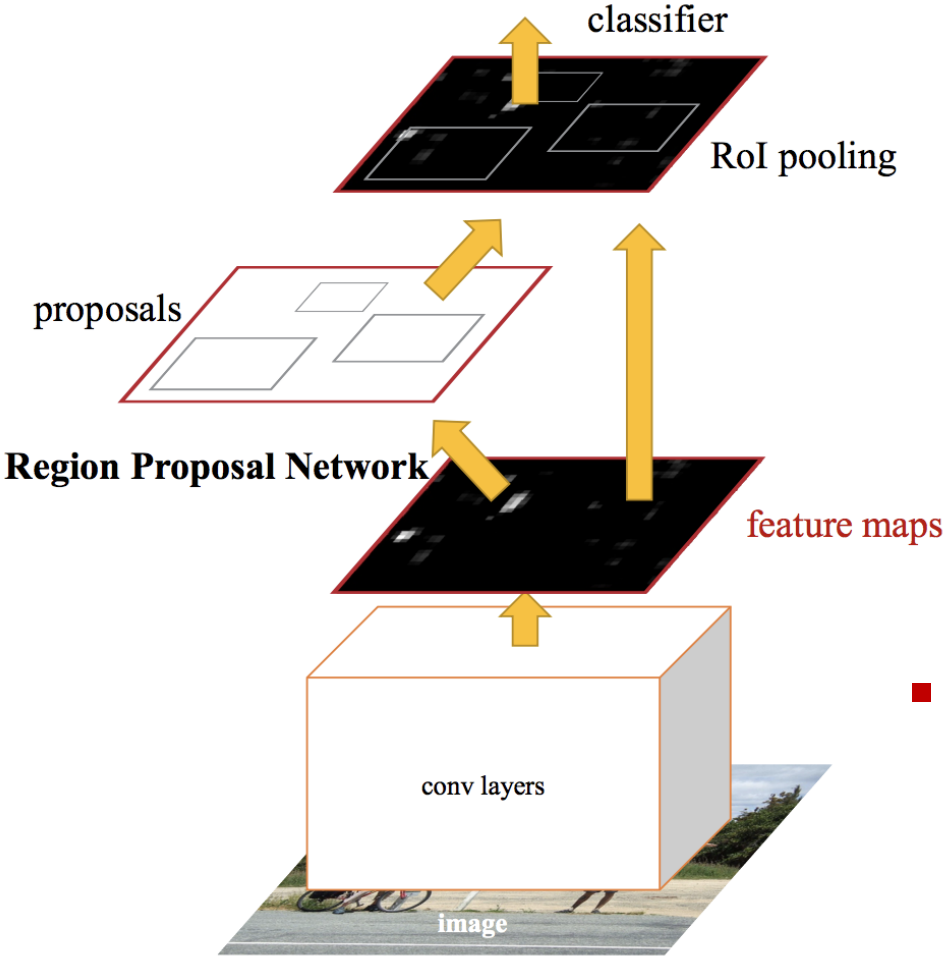
Source: github.com/kaichoulyc/tgs-salts

- Output = input dimension, fully convolutional

Encoder-decoder



Object detection = proposal + classification



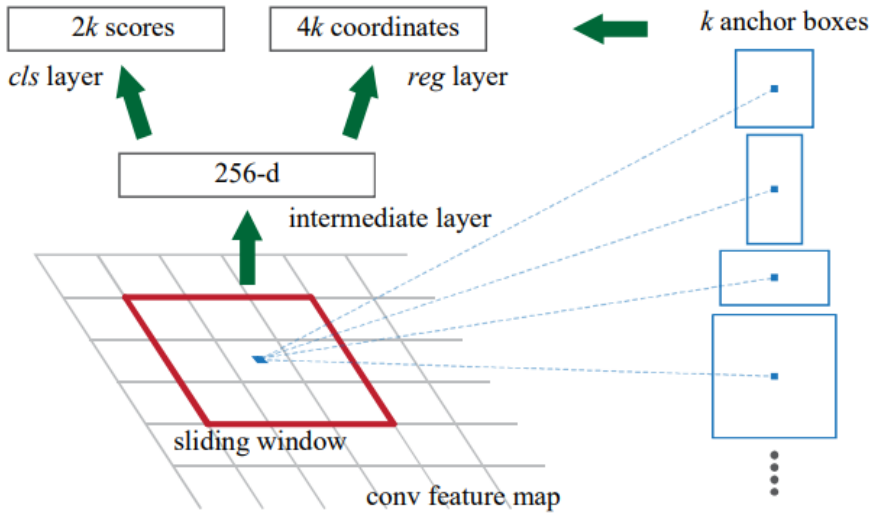
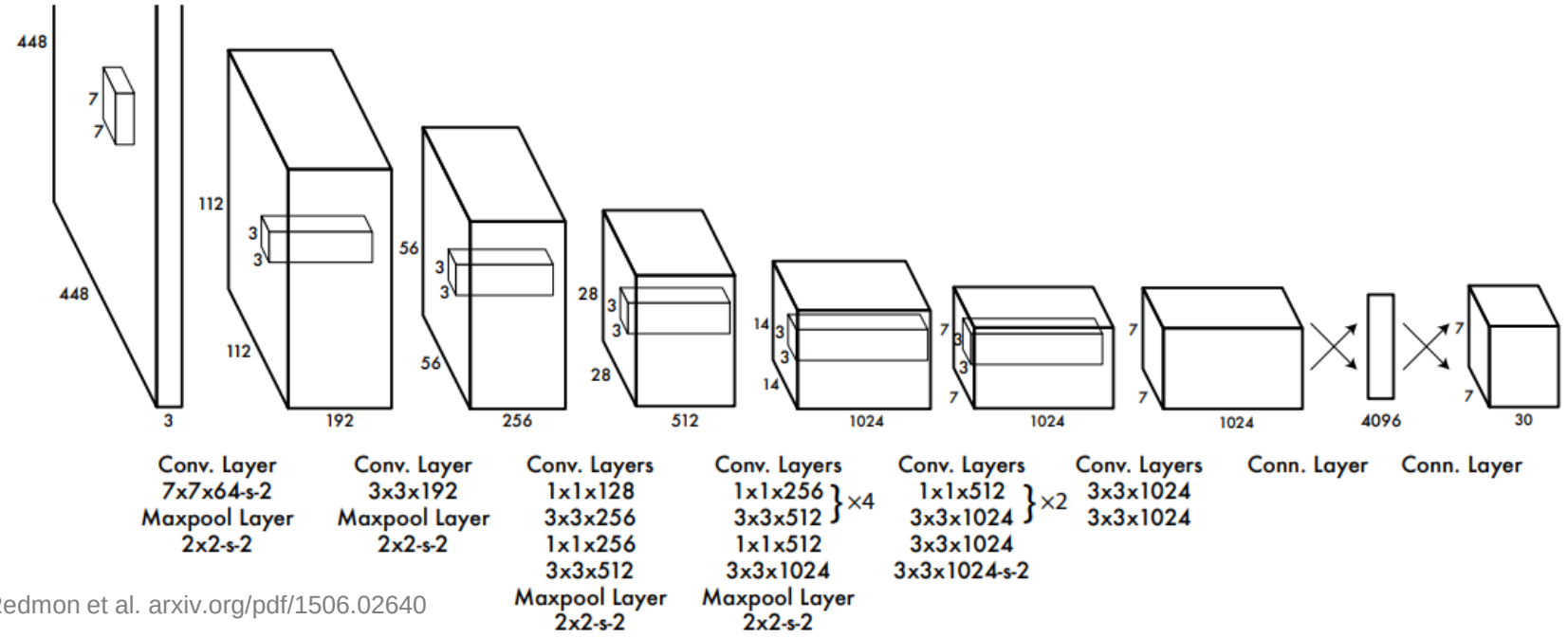


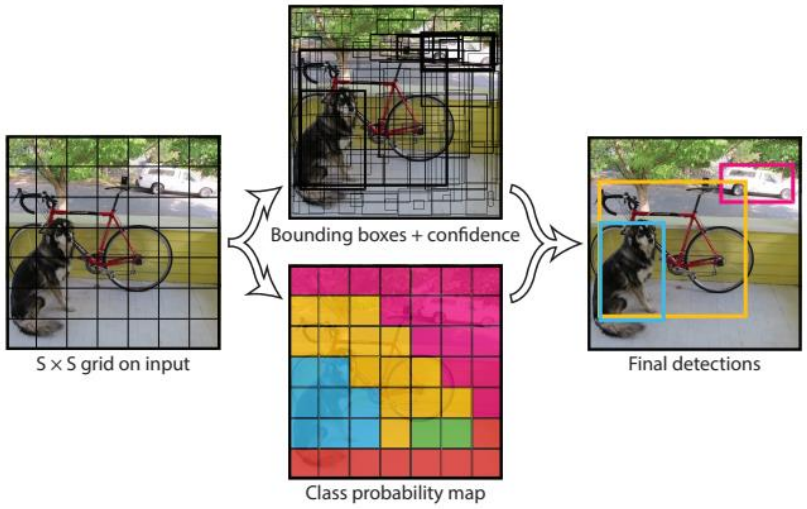
Image from towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

- Region proposal network is a CNN which predicts *k* bounding boxes as well as their confidence scores
- Each proposed region is then put into a classification model

YOLO – You only look once

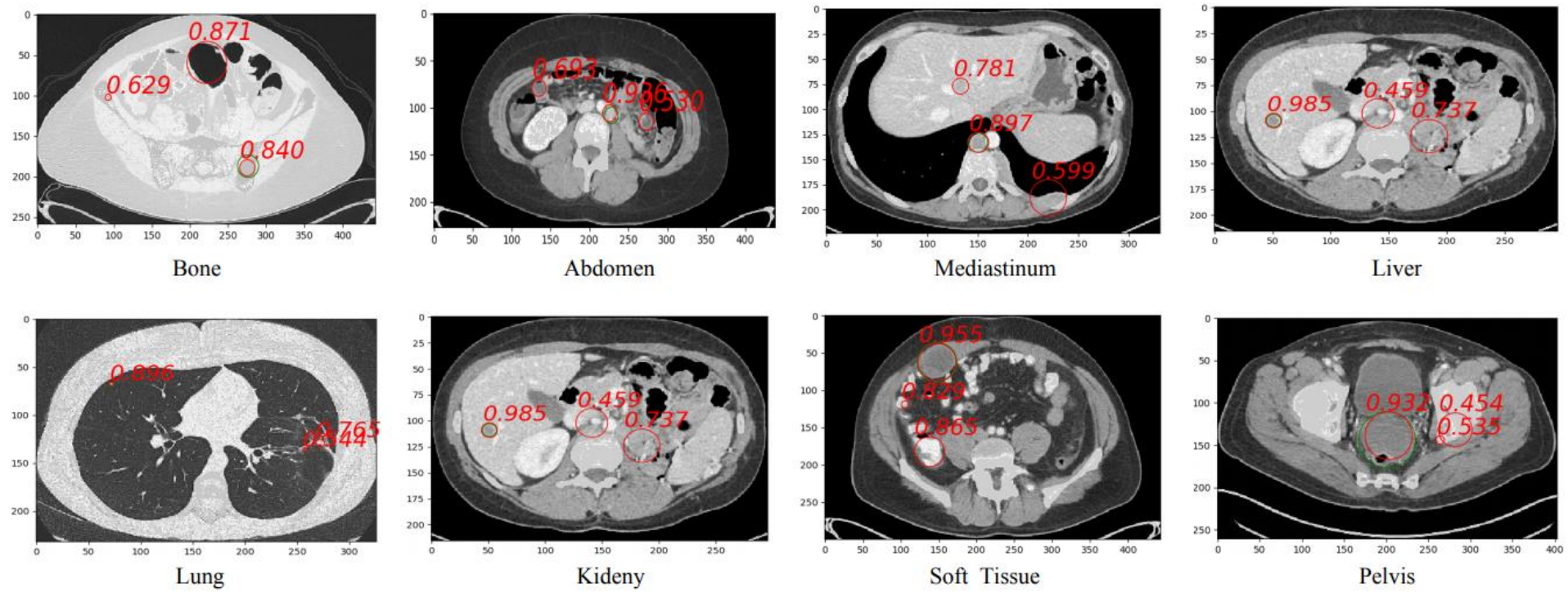


Redmon et al. arxiv.org/pdf/1506.02640



- Predict several SxS matrices
  - Bounding box locations
  - Bounding box confidence scores
  - Class confidence scores
- Single evaluation in CNN
- 45-155 FPS

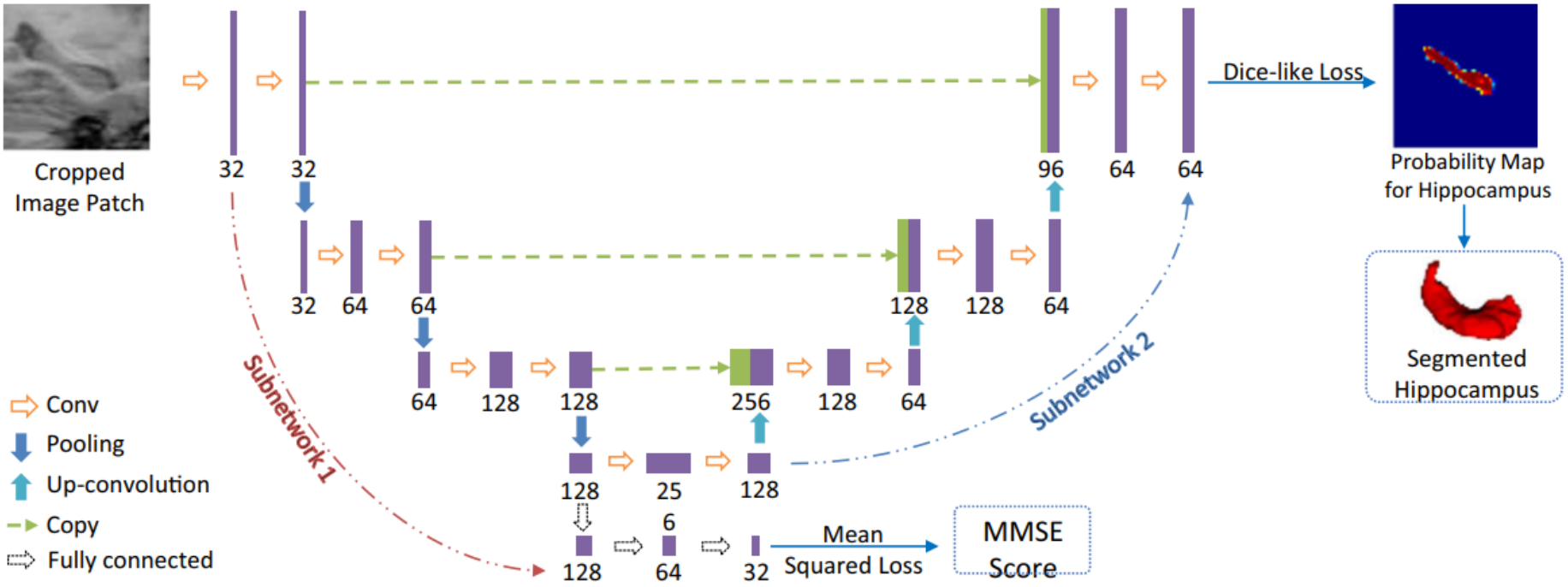
Lesion detection on CT images



Zhang et al. “3D Aggregated Faster R-CNN for General Lesion Detection” arxiv.org/pdf/2001.11071

- Propose possible lesion locations followed by lesion vs non-lesion classification

Joint segmentation and score prediction

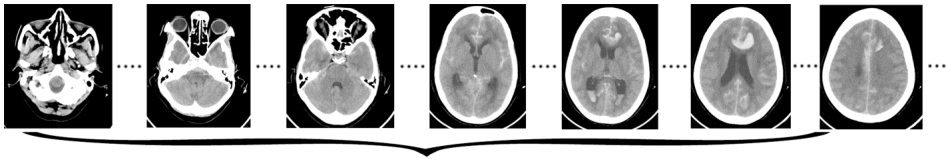


Cao et al. Multimed Tools Appl (2017)

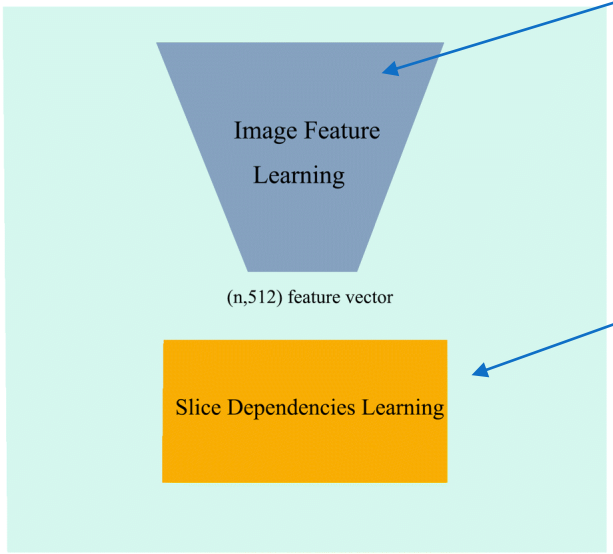
- U-Net with additional dense layers output for MMSE score prediction at the bottom of the convolutional network

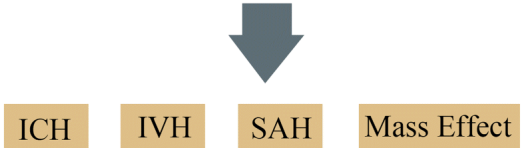
Handling multi-slice imaging data

Variable number of input slices



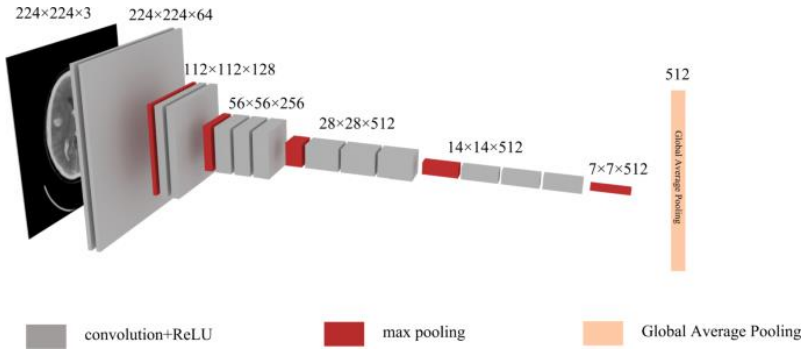
n images

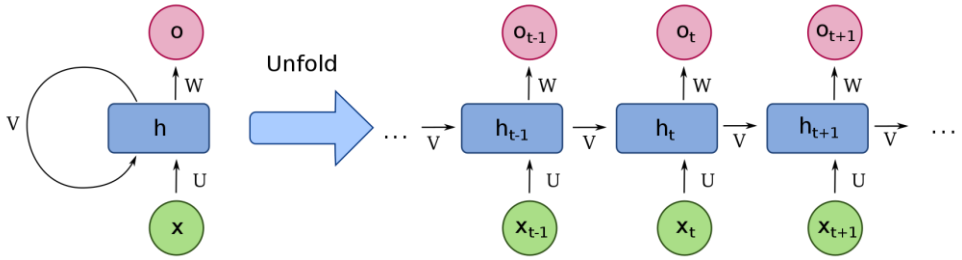




Li et al. BMC Bioinformatics (2020)

VGG16



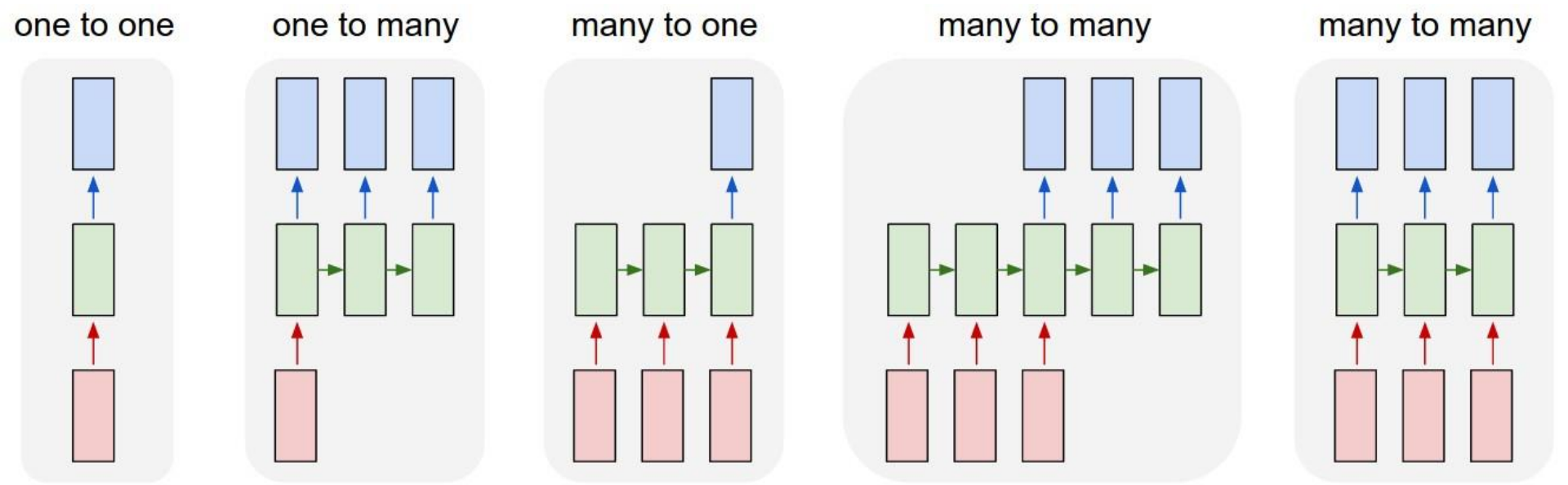


Recurrent Neural Network

- Most model requires a fixed number of inputs

Recurrent Neural Network

Applications of RNN



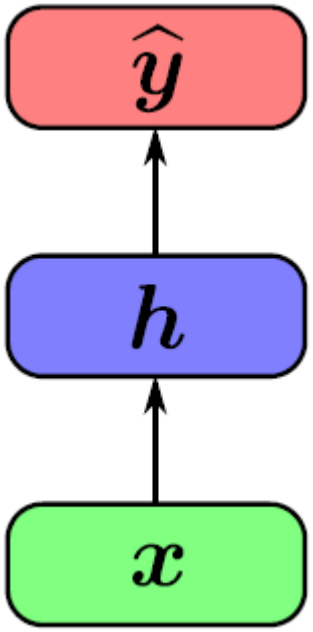
- Generate medical report = variable output length
- Extract keywords from written report = variable input length
- Translate sentence = variable input and output lengths

Formulation of RNN

Shared weights!

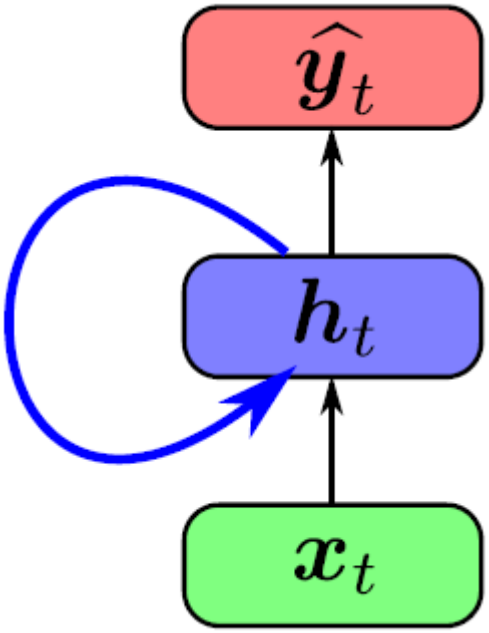
$$h_1 = f(\boldsymbol{u} \cdot x_1 + \boldsymbol{v} \cdot h_0 + c)$$
$$h_2 = f(\boldsymbol{u} \cdot x_2 + \boldsymbol{v} \cdot h_1 + c)$$
$$\dots$$
$$h_t = f(\boldsymbol{u} \cdot x_t + \boldsymbol{v} \cdot h_{t-1} + c)$$
$$\hat{y}_t = \boldsymbol{w} \cdot h_t + b$$

$$h = f(\boldsymbol{u} \cdot x + c)$$
$$\hat{y} = \boldsymbol{w} \cdot h + b$$



Output of a hidden layer

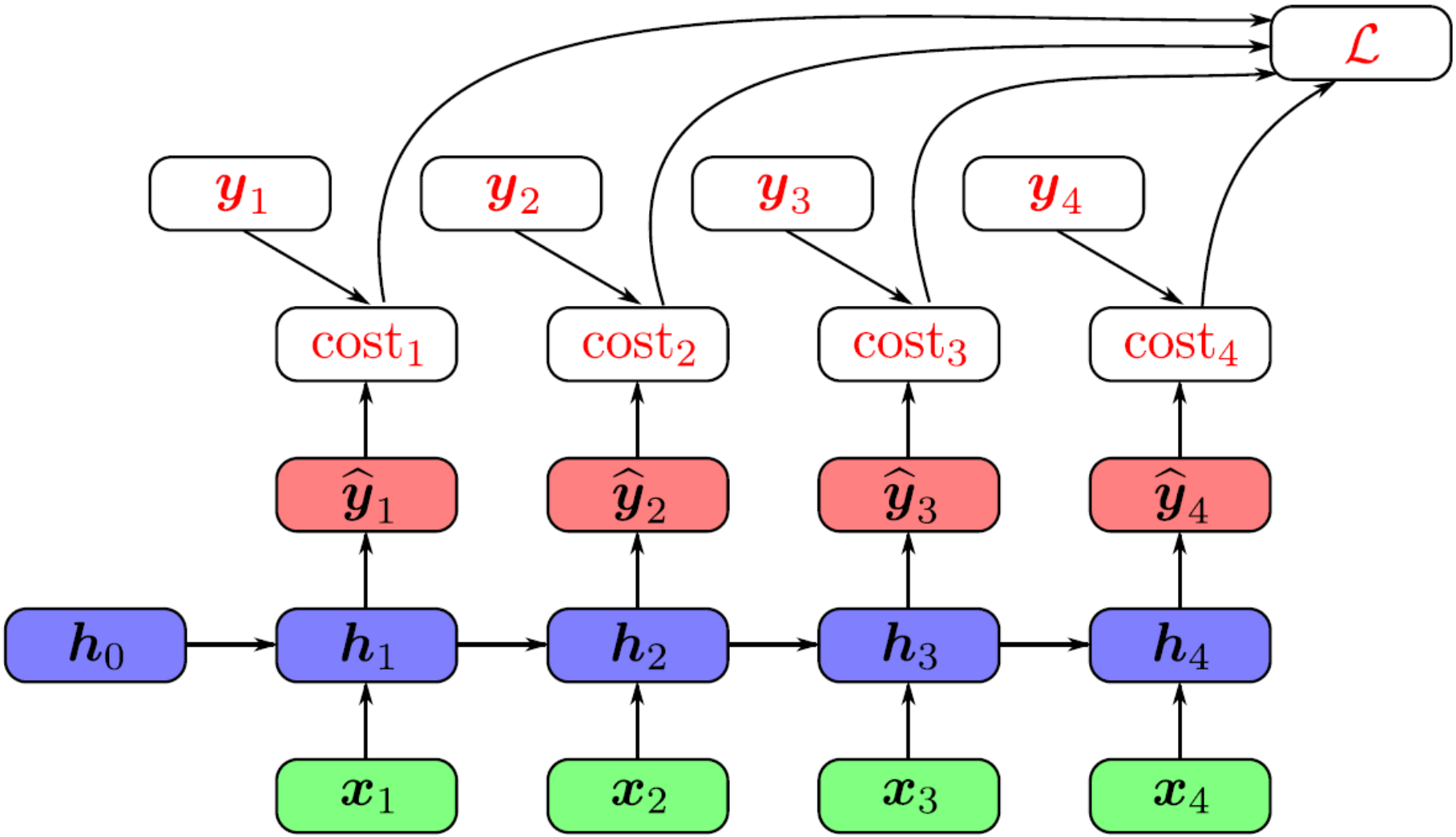
Fixed-length input



Output of a hidden layer

Variable-length input

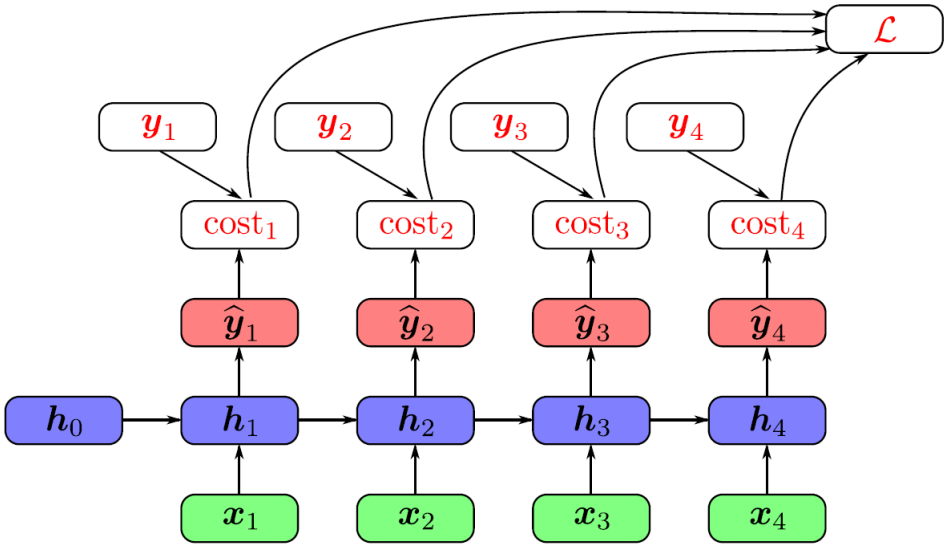
Gradient routes in RNN



- Loss function is evaluated at the end of the input sequence

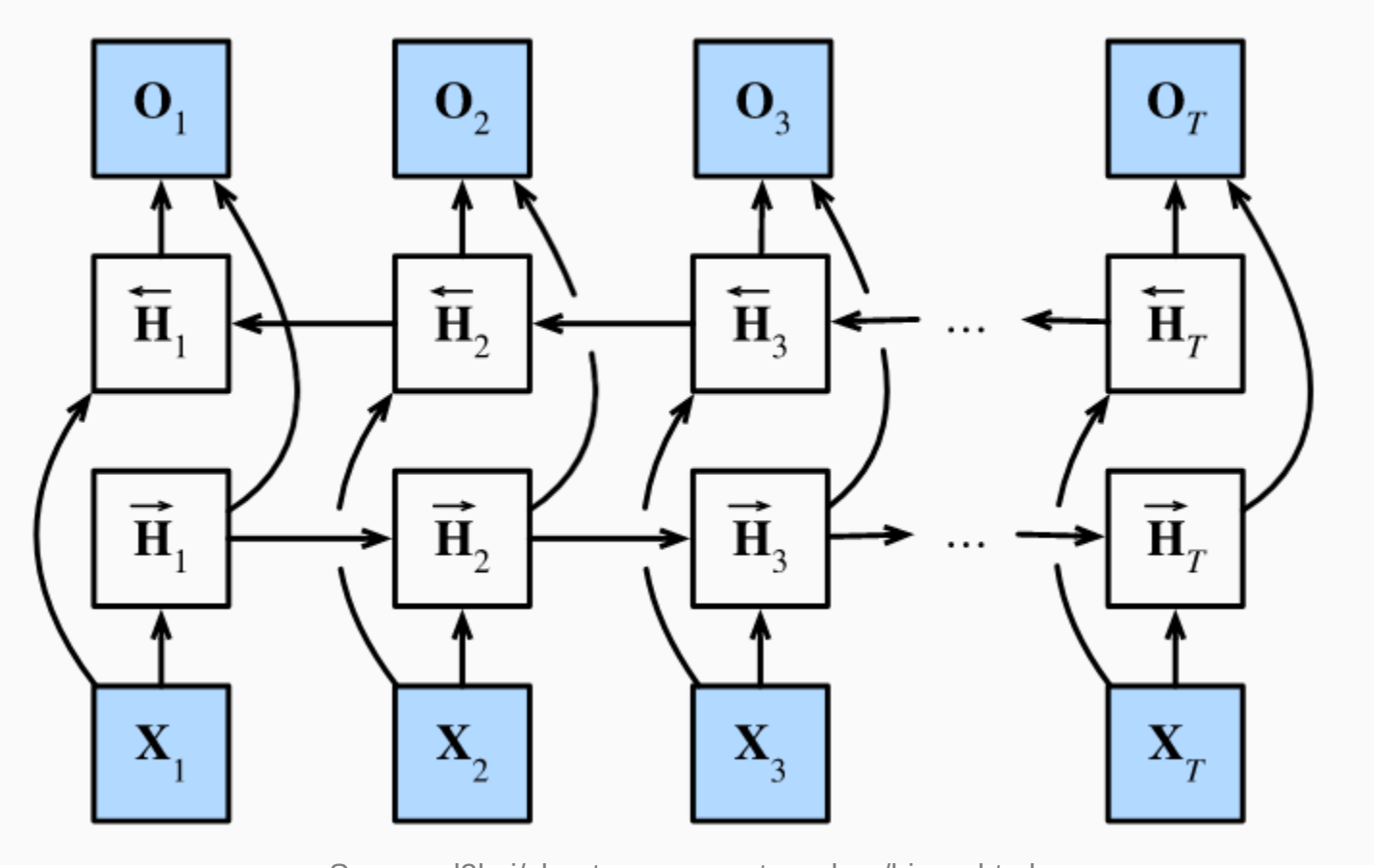
Backpropagation through time

$$h_1 = f(\boldsymbol{u} \cdot x_1 + \boldsymbol{v} \cdot h_0 + c)$$
$$h_2 = f(\boldsymbol{u} \cdot x_2 + \boldsymbol{v} \cdot h_1 + c)$$
$$\dots$$
$$h_t = f(\boldsymbol{u} \cdot x_t + \boldsymbol{v} \cdot h_{t-1} + c)$$
$$\hat{y}_t = \boldsymbol{w} \cdot h_t + b$$



- Because weights $\boldsymbol{u}$ and $\boldsymbol{v}$ are shared, we backpropagate gradients through all h_i
  - $\frac{\delta L}{\delta u} = \frac{\delta L}{\delta h_1}\frac{\delta h_1}{\delta u} + \frac{\delta L}{\delta h_2}\frac{\delta h_2}{\delta u} + \frac{\delta L}{\delta h_3}\frac{\delta h_3}{\delta u} + \frac{\delta L}{\delta h_4}\frac{\delta h_4}{\delta u}$
  - $\frac{\delta L}{\delta v} = \frac{\delta L}{\delta h_1}\frac{\delta h_1}{\delta v} + \frac{\delta L}{\delta h_2}\frac{\delta h_2}{\delta v} + \frac{\delta L}{\delta h_3}\frac{\delta h_3}{\delta v} + \frac{\delta L}{\delta h_4}\frac{\delta h_4}{\delta v}$

Bidirectional model

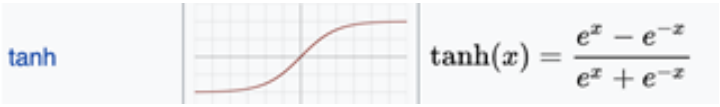


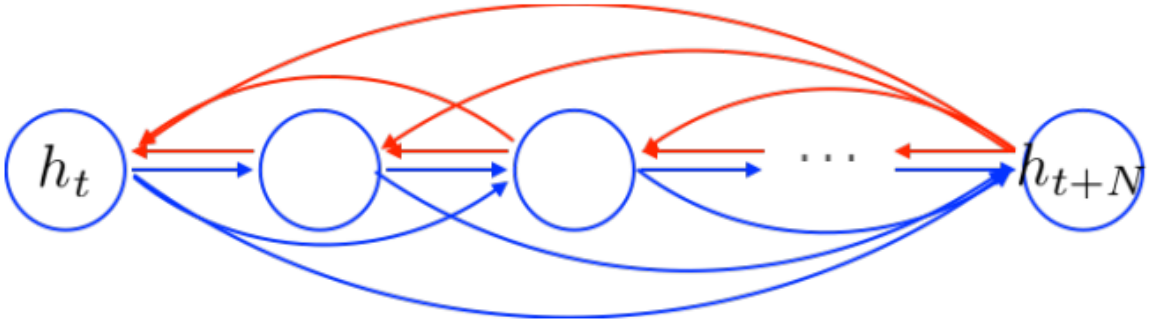
Source: d2l.ai/chapter_recurrent-modern/bi-rnn.html

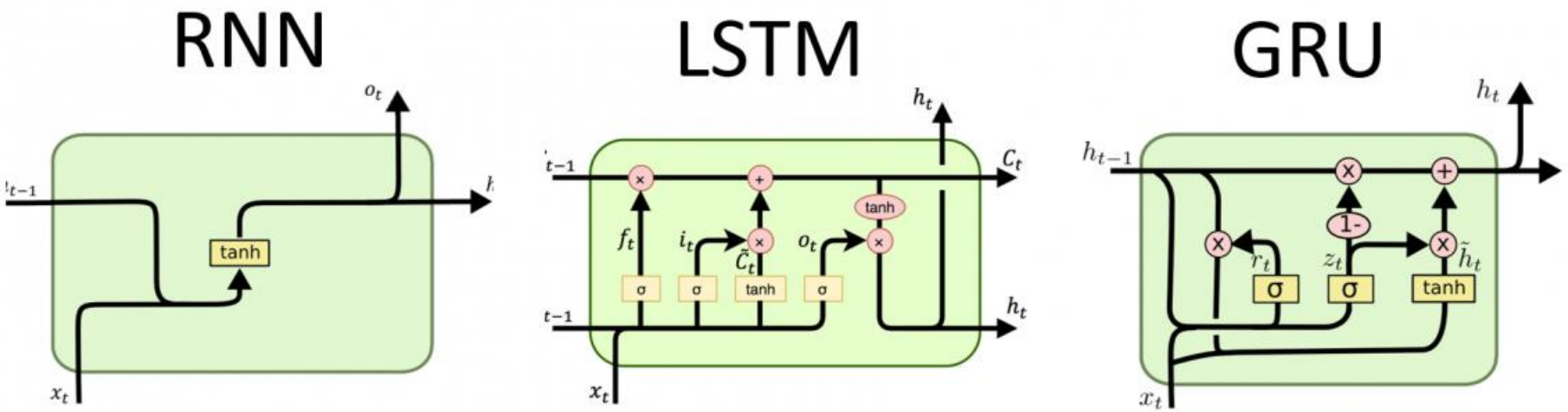
- Just as RNN can learn h_t from h_{t-1} and x_t, it can also learn h_{t-1} from h_t and x_{t-1}
- Combine information from the forward and backward passes

RNN architectures

Adding “gates” to RNN







Source: www.linkedin.com/pulse/recurrent-neural-networks-rnn-gated-units-gru-long-short-robin-kalia

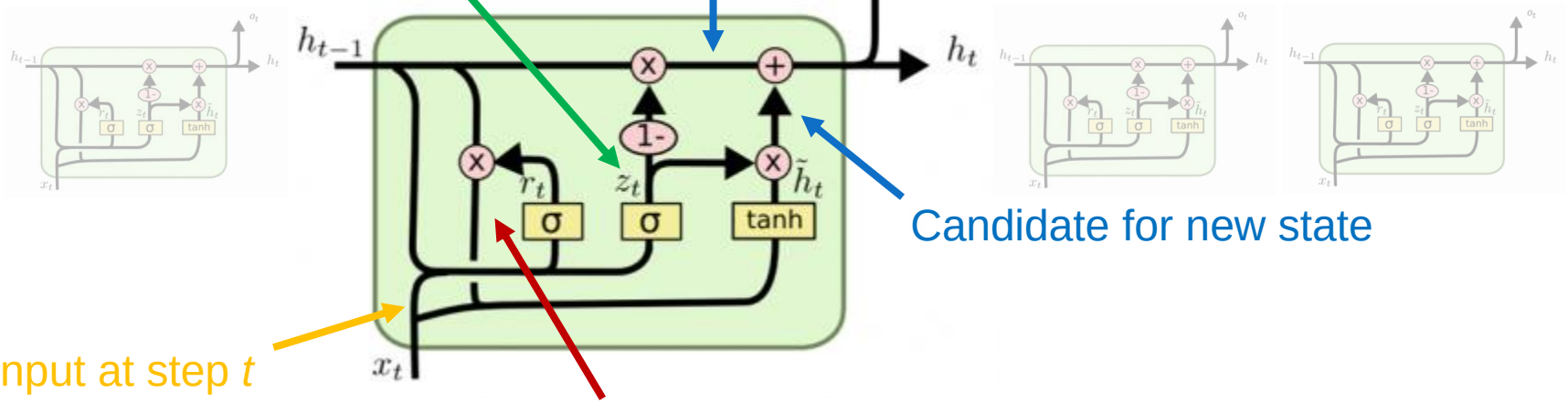
- Add a direct path from previous unit to the next
  - Allow the network to “forget” the past or “remember” it
  - h_t can be updated with x_t or stay the same as h_{t-1}

Gated Recurrent Unit (GRU)

Update Gate: Weight for keeping previous state or updating to the new state

Previous state

Output at step *t*



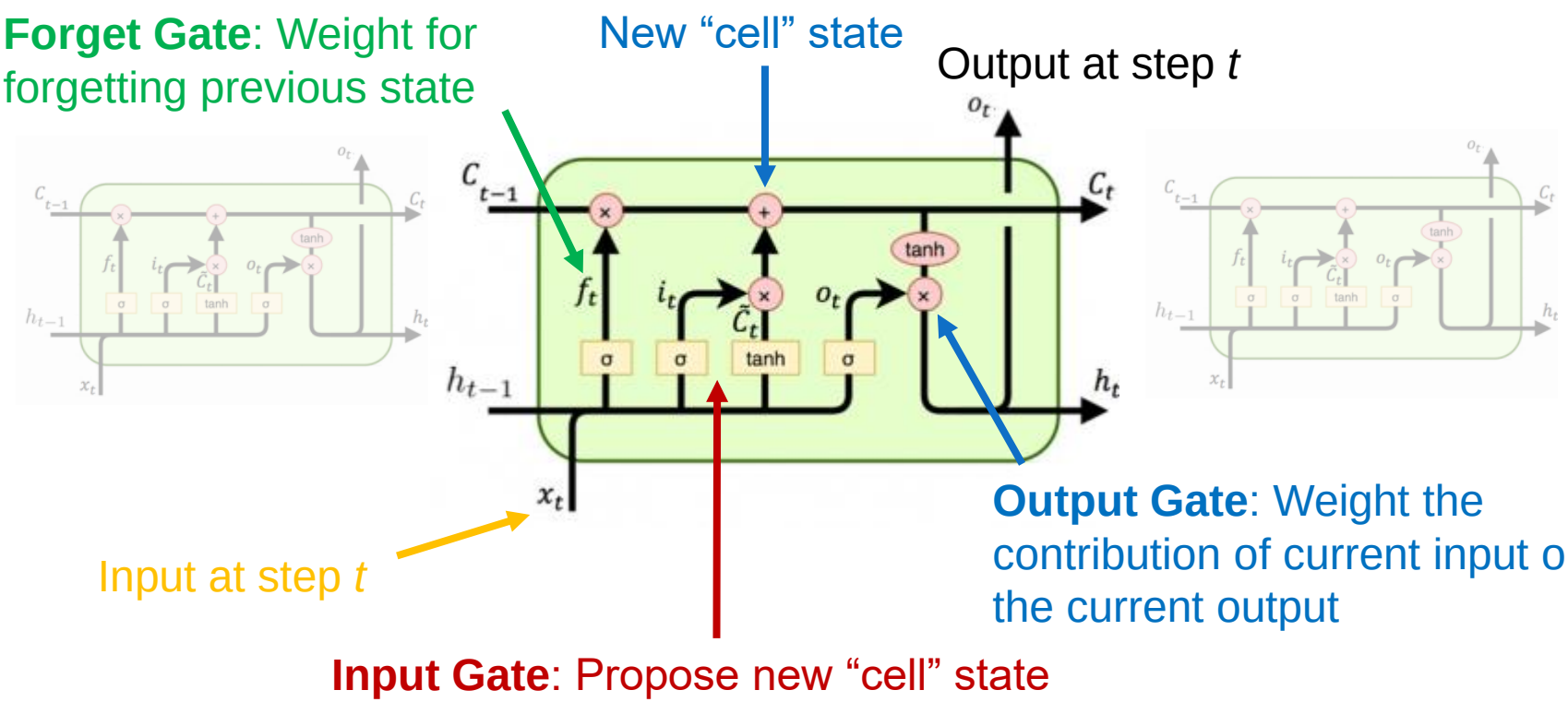
Candidate for new state

Input at step *t*

Reset Gate: Weight for whether to consider previous state when proposing the new state

- New input, together with previous state, is used to determine whether to remember or forget the previous state
- Gradient can flow through the $h_{t-1} \rightarrow h_t$ shortcut

Long Short-Term Memory (LSTM)



- Cell state C_t can carry memory over long term
- The contribution of current input on the output at step *t* can be assigned through the Output Gate

Applications of RNN

RNN for variable multi-slices input

Variable number of input slices

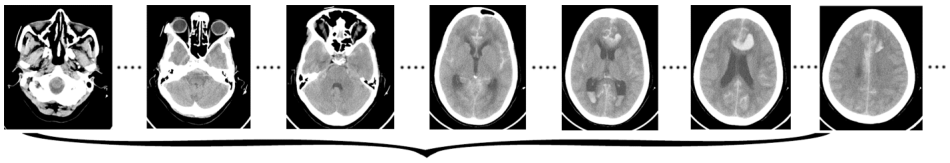
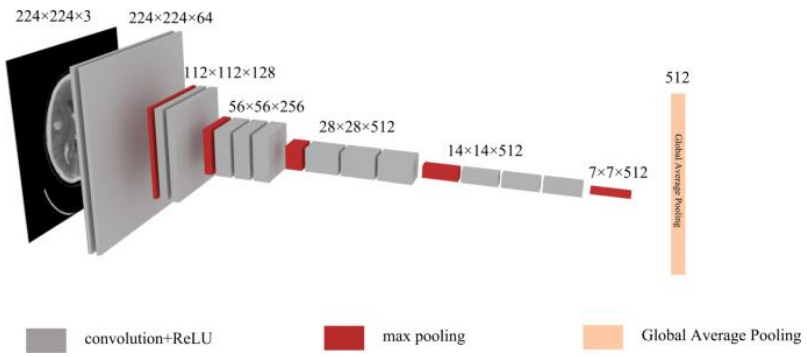
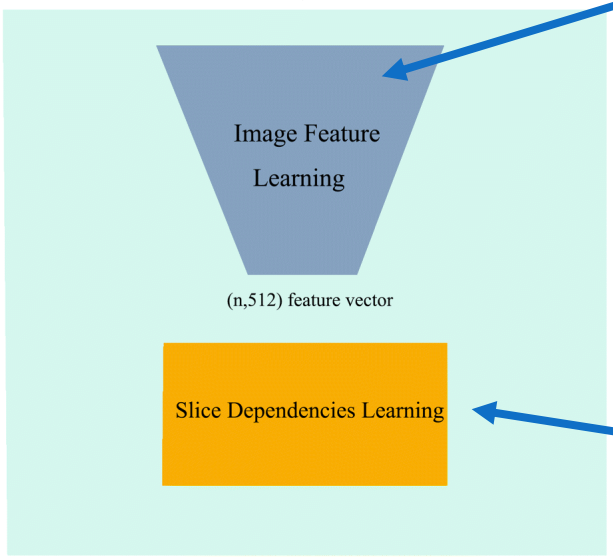
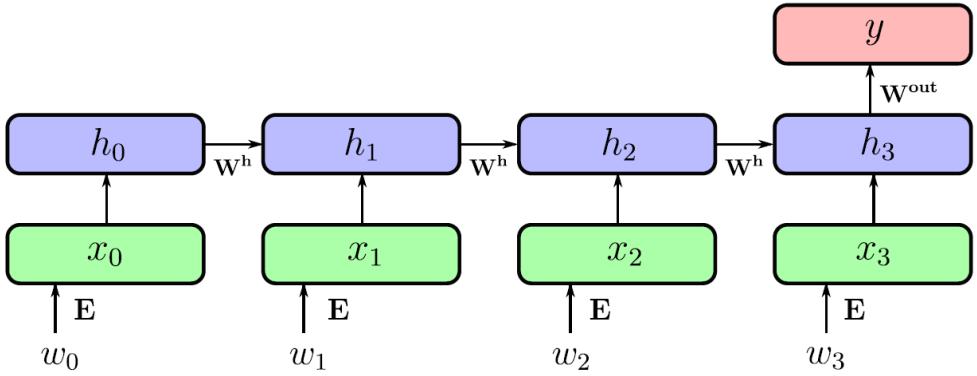


Image encoder





Regression output

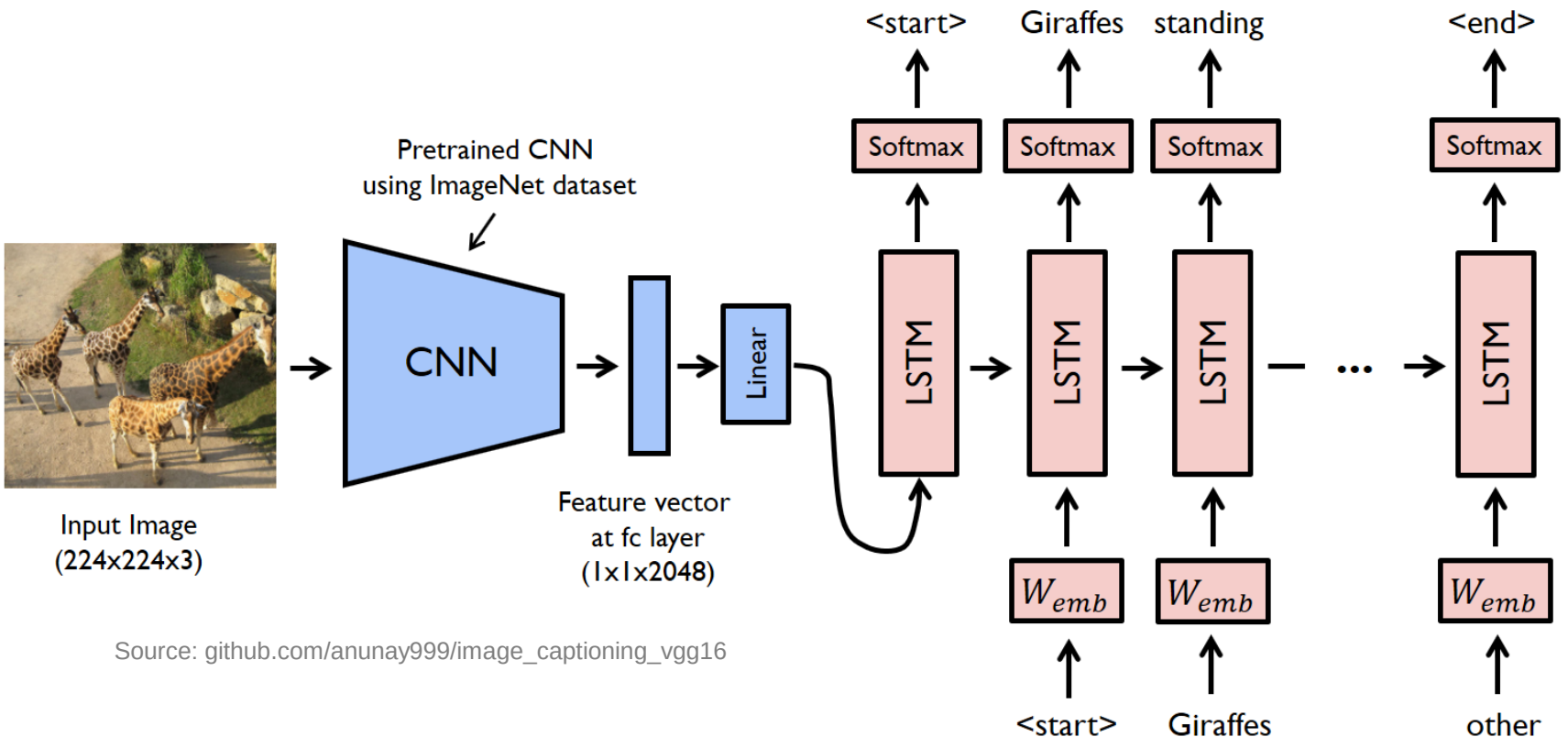


Embeddings of input images

ICH IVH SAH Mass Effect

Li et al. BMC Bioinformatics (2020)

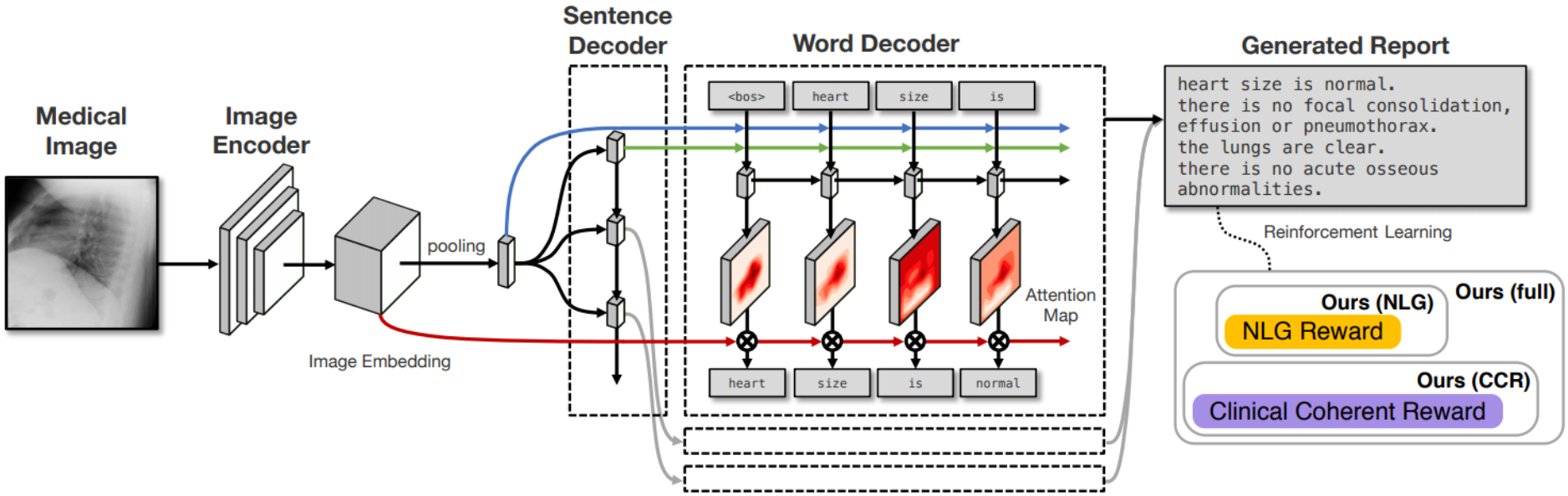
Image captioning



Source: github.com/anunay999/image_captioning_vgg16

- Embedding of input image is sent into an RNN to generate variable-length output sentence(s)
- Each output is a probability distribution over all words

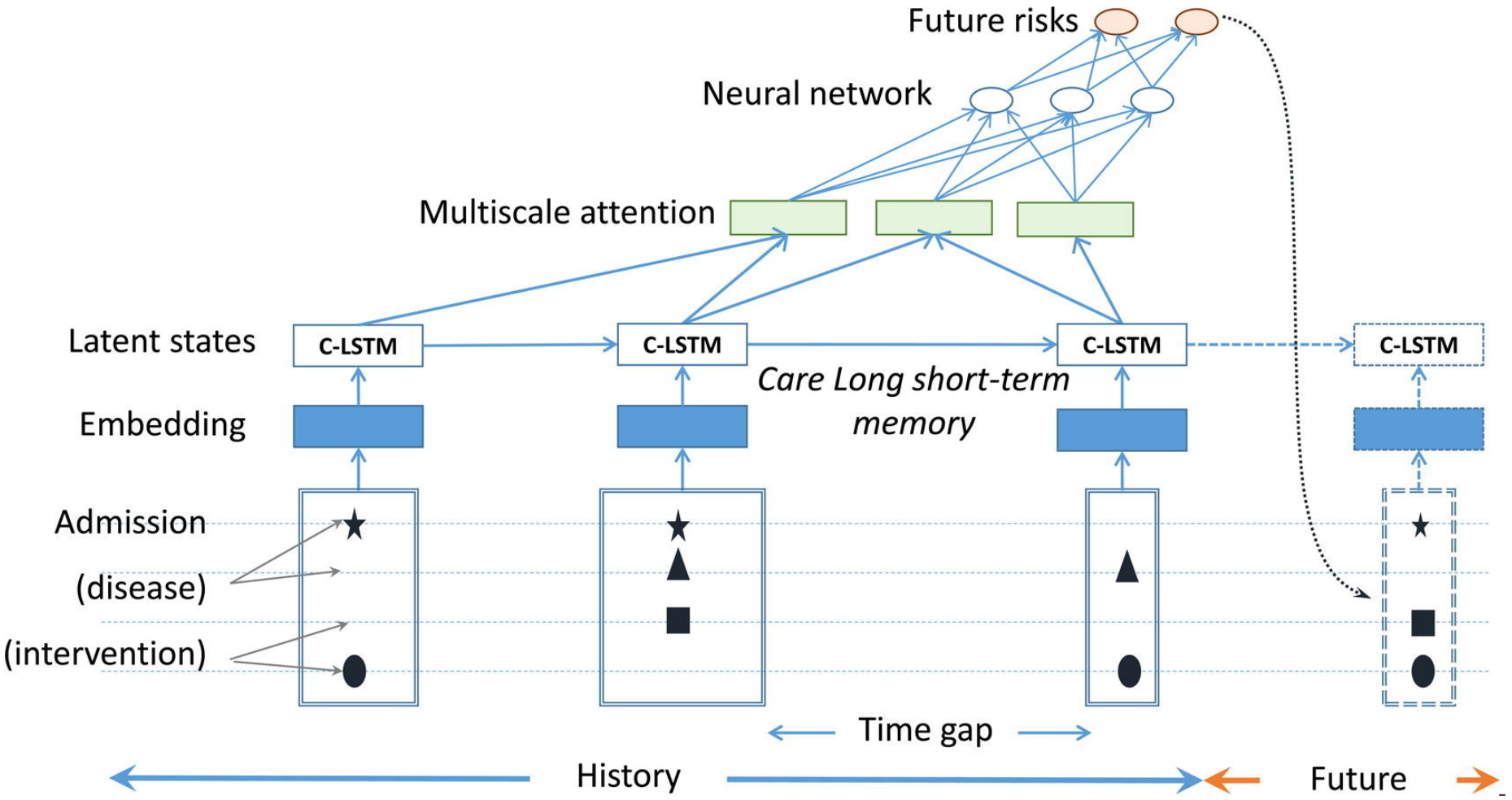
Medical report generation



Liu et al. Clinically Accurate Chest X-Ray Report Generation (2019)

- Same principle as image captioning
- **Sentence decoder** generate “topic” + “stop token”
  - **Word decoder** then generate words that fill each sentence according to the predict “topic”
- 2D embedding of the input image is fed to the word decoder to relate each predicted word with location in input image

RNN for EHR-based prediction



Pham et al. J of Biomed Informatics (2017)

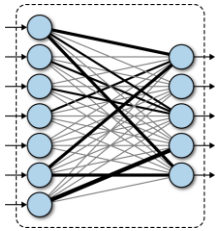
- Embed medical terms and feed time-series EHR to an RNN

Unsupervised learning with ANN “Autoencoder”

Embedding words

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Image from hackermoon.com



cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8

man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

Image from medium.com

- Word encoder = a fully connected neural network
  - Input = one-hot encoding of words
  - Output = a continuous-value embedding
  - Used as encoder for a larger network (transfer learning)

Word embedding based on context

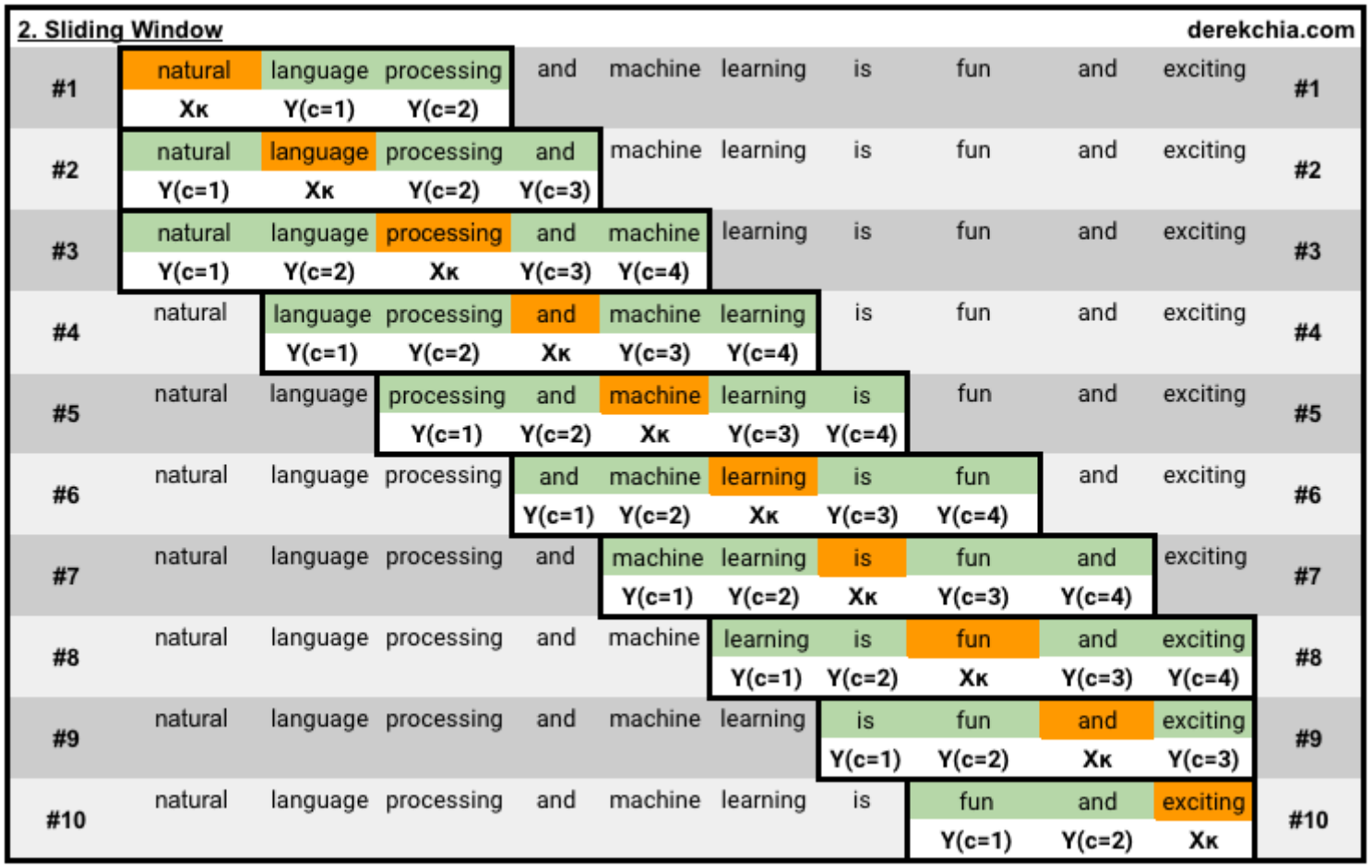


Image from towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

- Embedding of adjacent words should be similar
- Embedding should be able to predict context relationship

Skip-gram model

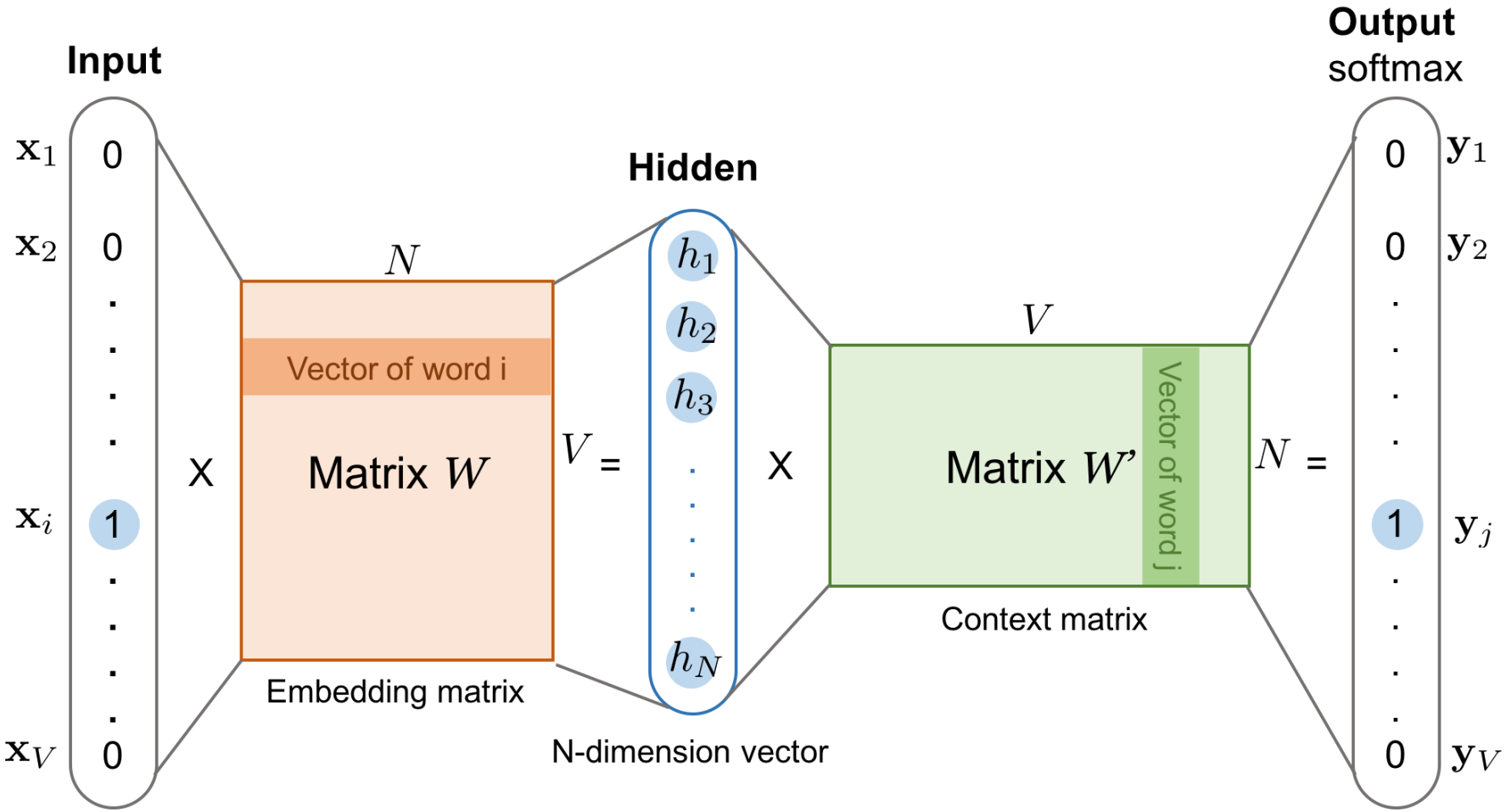
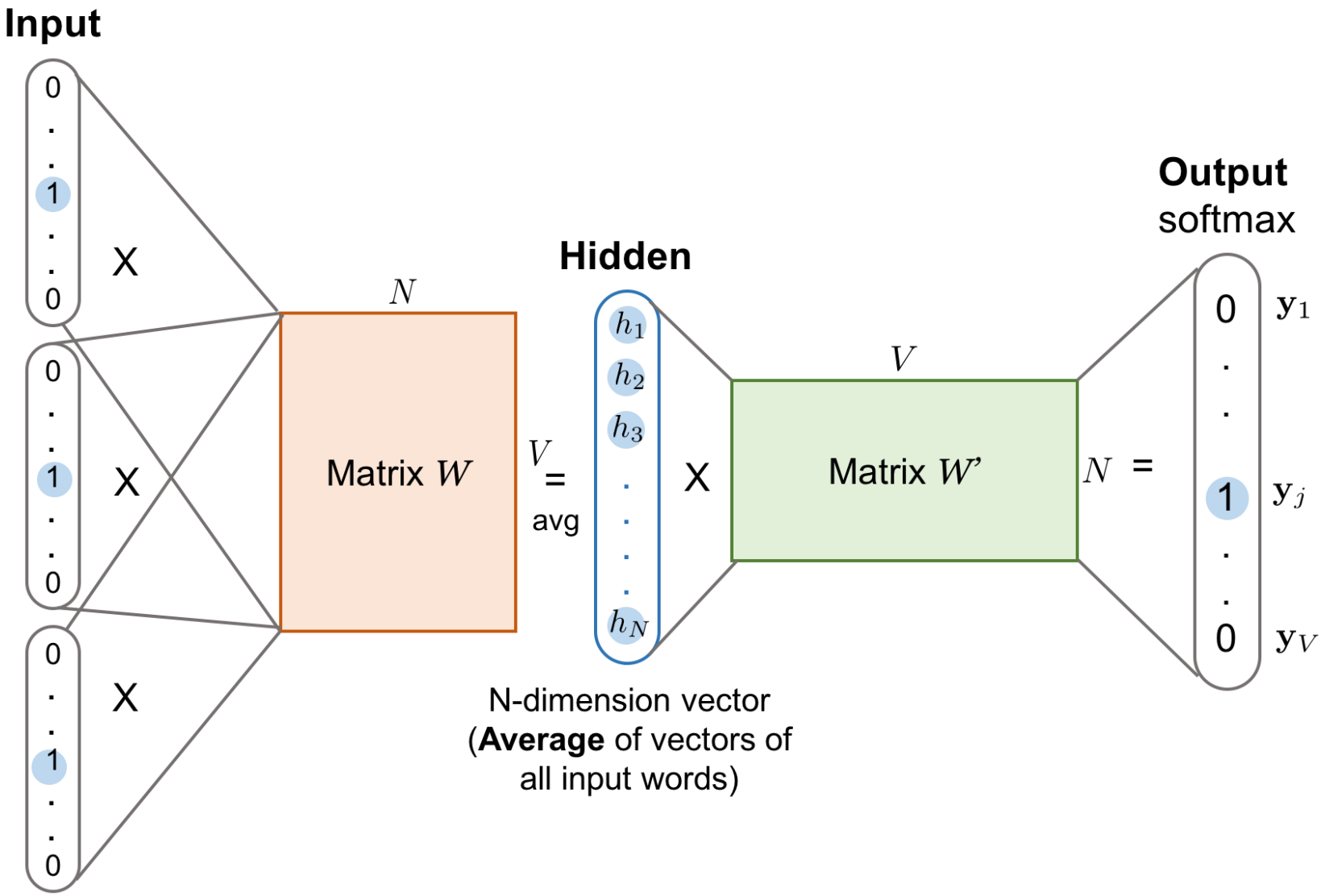


Image from towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

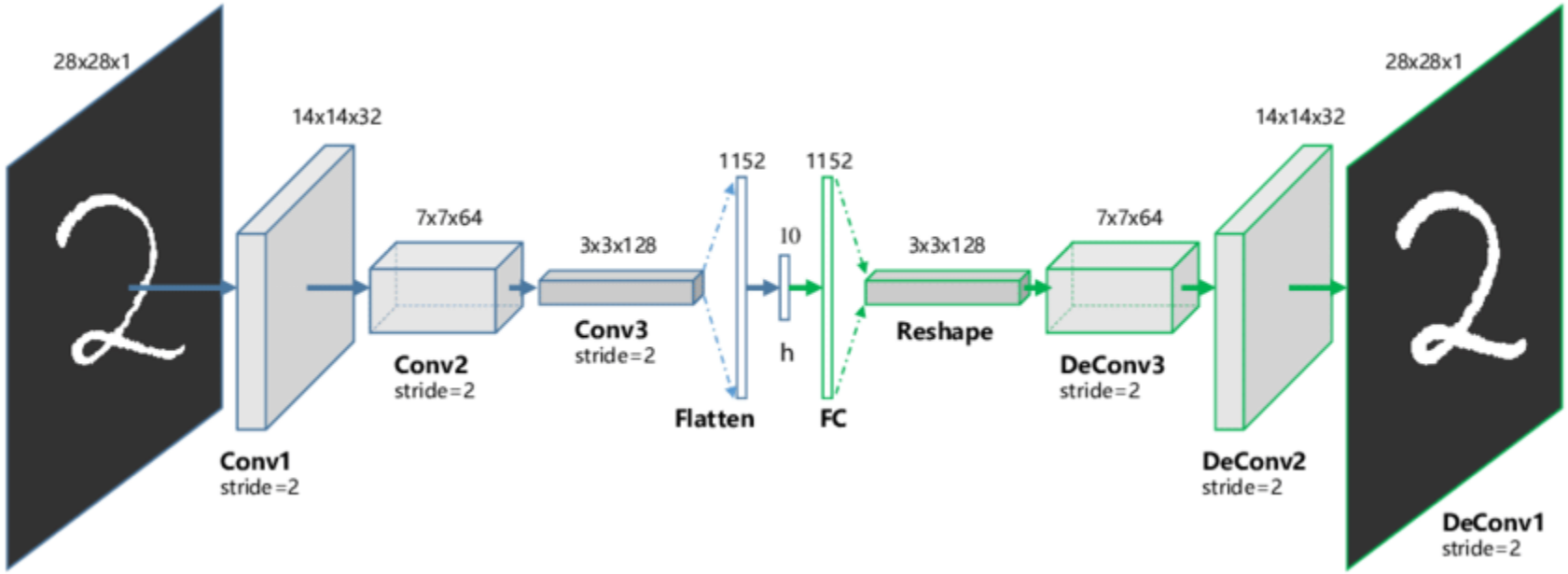
- Predict context words from an input word
  - Training data = pair of context words

Continuous bag-of-words model



- Predict center word from average embeddings of context words

Autoencoder



Guo et al. International Conference on Neural Information Processing (2017)

- Train the model to predict input data themselves
- But squeeze the embedding dimension (output of the flatten layer above) to prevent the model from just “remembering”
  - This is called “bottleneck”
- This also reproduced all noises and errors

Denoising autoencoder

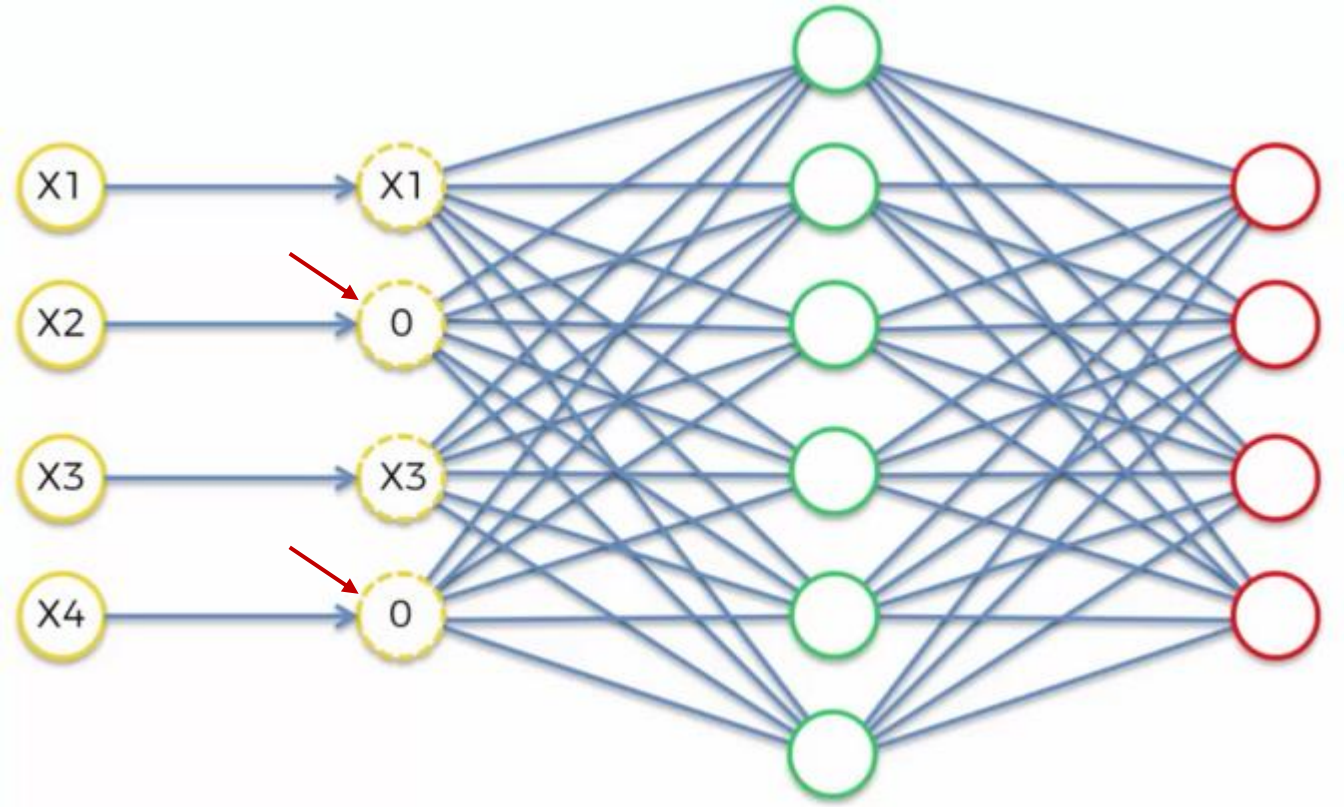


Image from towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2

- Introduce noises into the input before feeding into the model
- But calculate loss based on original input
  - Prevent model from overfitting to noises and errors in input
  - Similar to Dropout

Variational autoencoder (VAE)

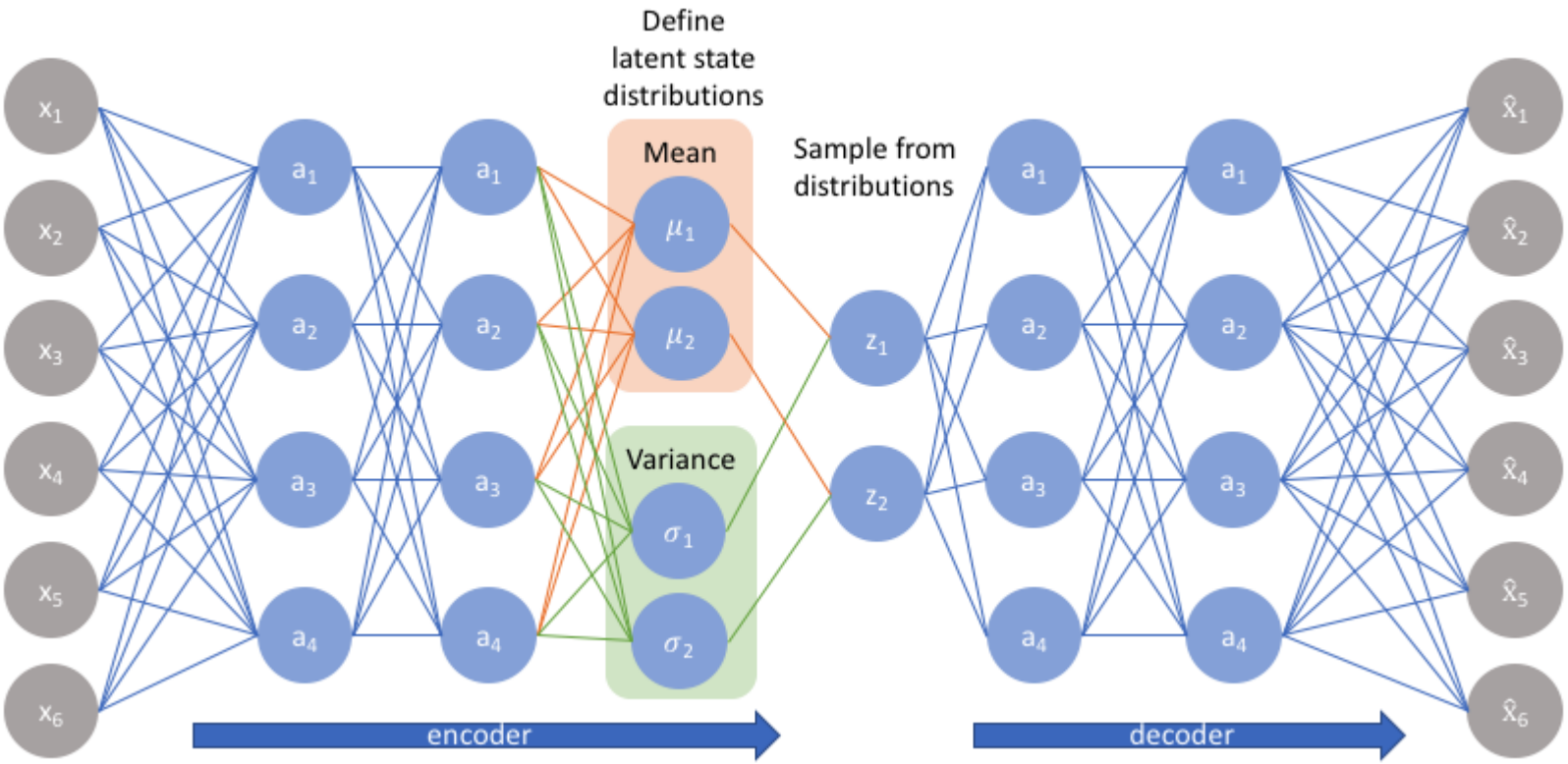


Image from www.jeremyjordan.me/variational-autoencoders/

- Embed the mean and variance
- Sample from Normal distribution for the decoder
  - $z_i \sim \text{Normal}(\mu_i, \sigma_i) = \mu_i + \sigma_i \times \text{Normal}(0, 1)$
  - Backpropagate through $z_i \rightarrow (\mu_i, \sigma_i)$

Challenges and techniques in deep learning

Gradient descent in multi-dimension

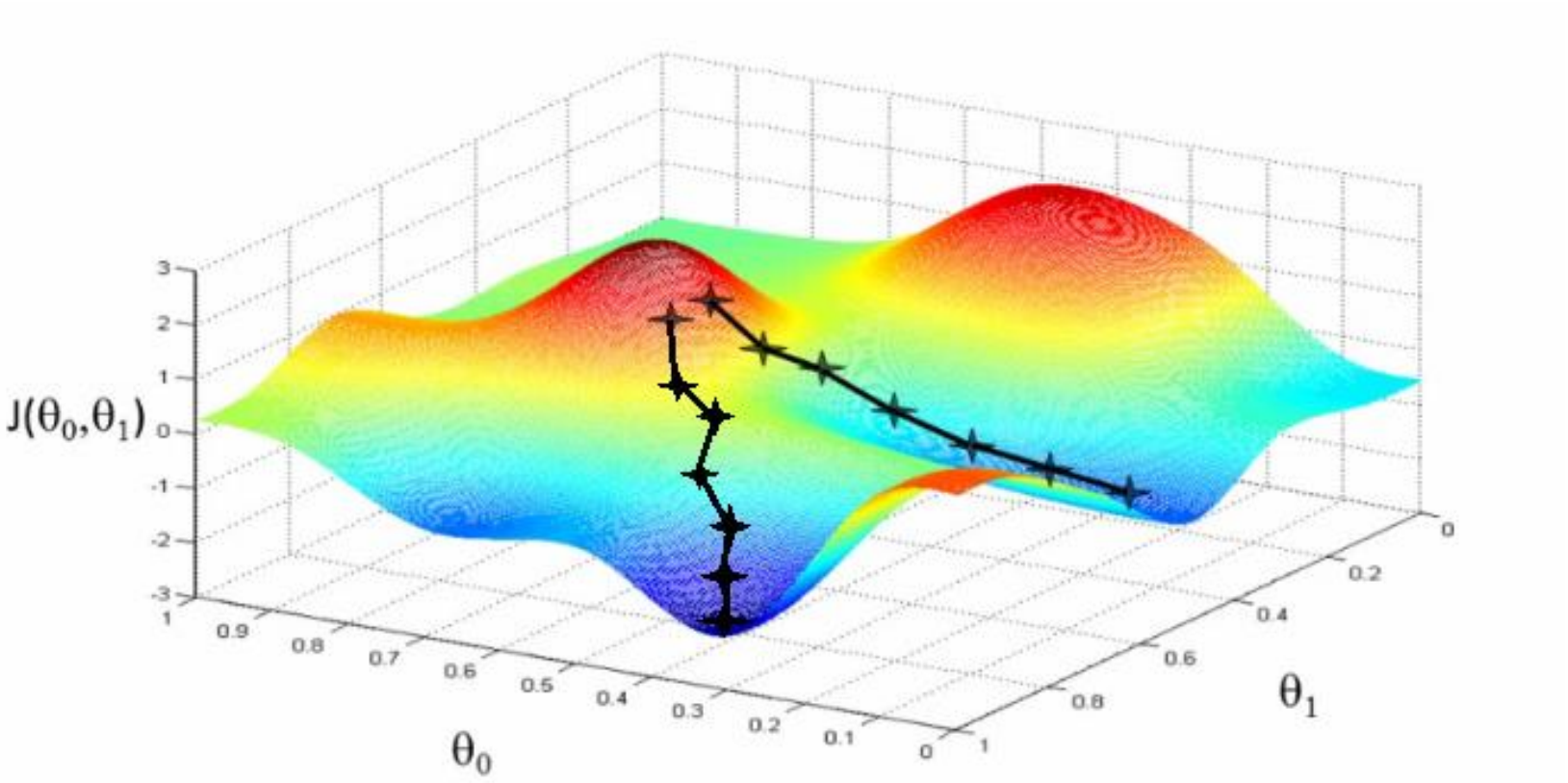


Image from shashank-ojha.github.io

- Starting position can determine which local minima to model converges to
  - When training artificial neural network model, we want to try several random initial weights

Gradient descent with momentum

Without momentum

With momentum

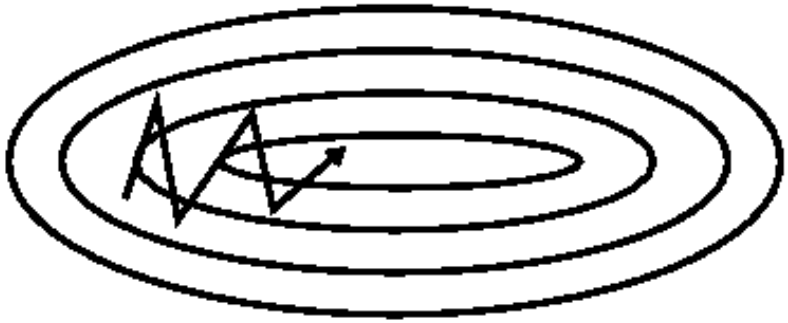
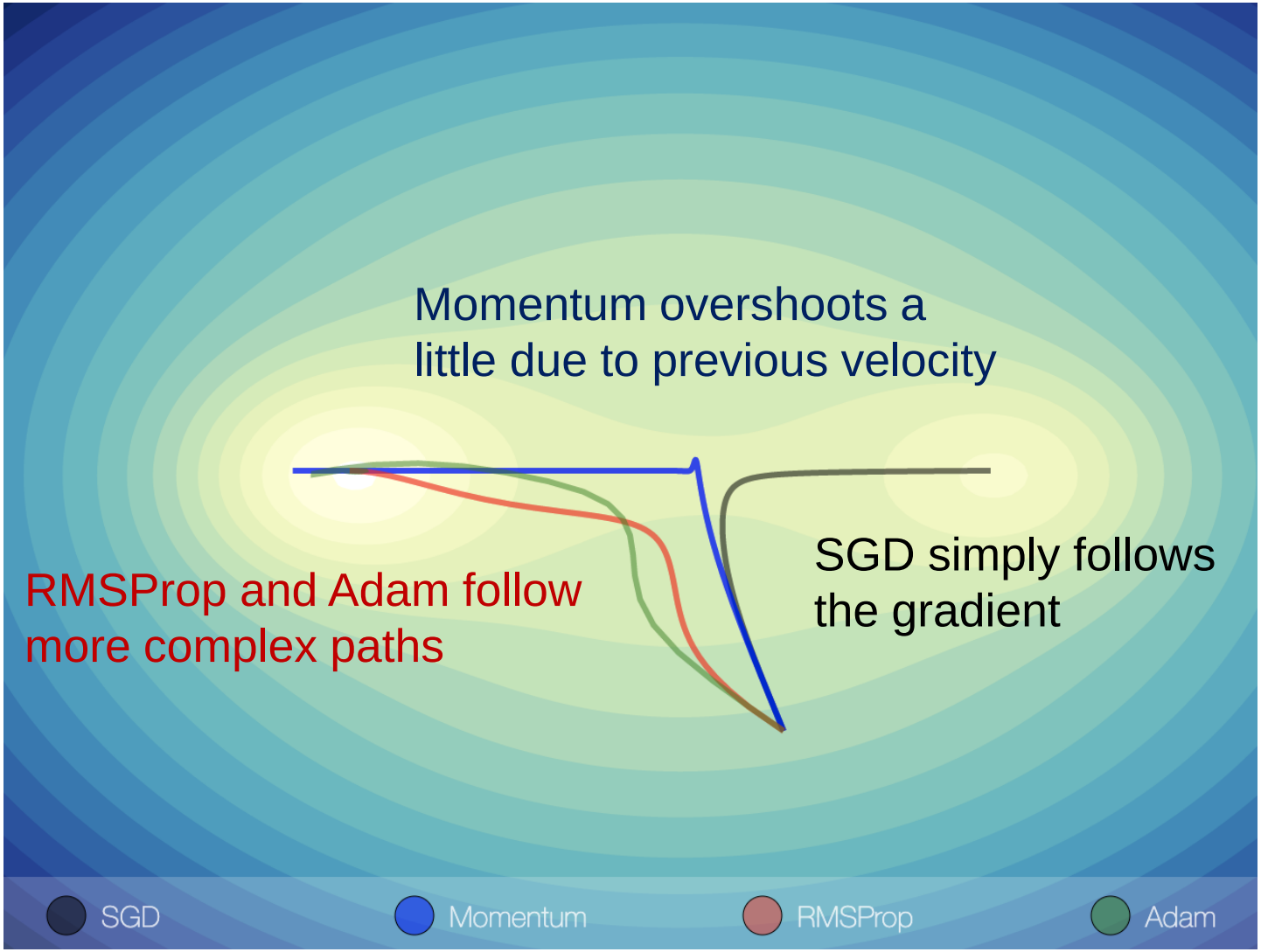


Image from ruder.io

- Gradient descent
  - $w^{t+1} = w^t - \eta \nabla_w (w^t; \text{data})$
  - No memory from previous update
- Momentum
  - Previous update (velocity) influence the next update (velocity)
  - $v^{t+1} = \gamma v^t + \eta \nabla_w (w^t; \text{data})$
  - $w^{t+1} = w^t - v^{t+1}$

Various optimizers



https://emiliendupont.github.io/2018/01/24/optimization-visualization/

Cyclical learning rate

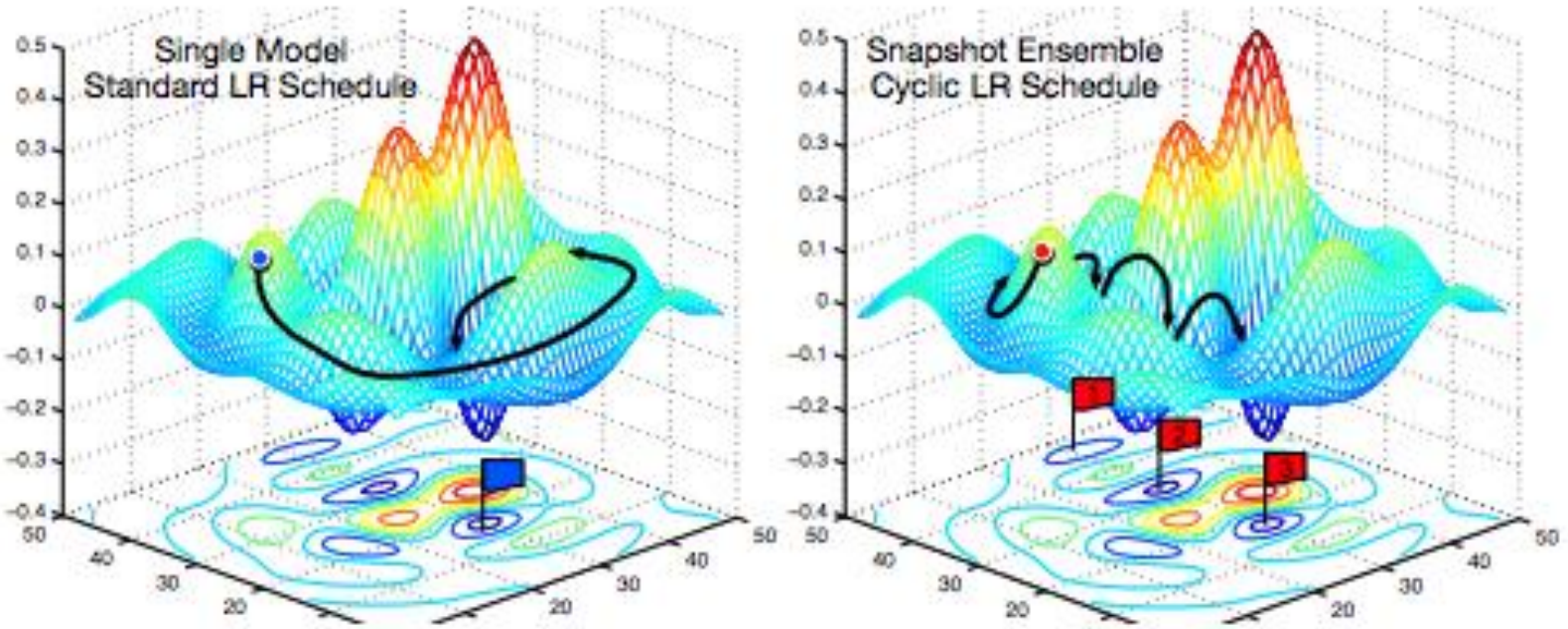
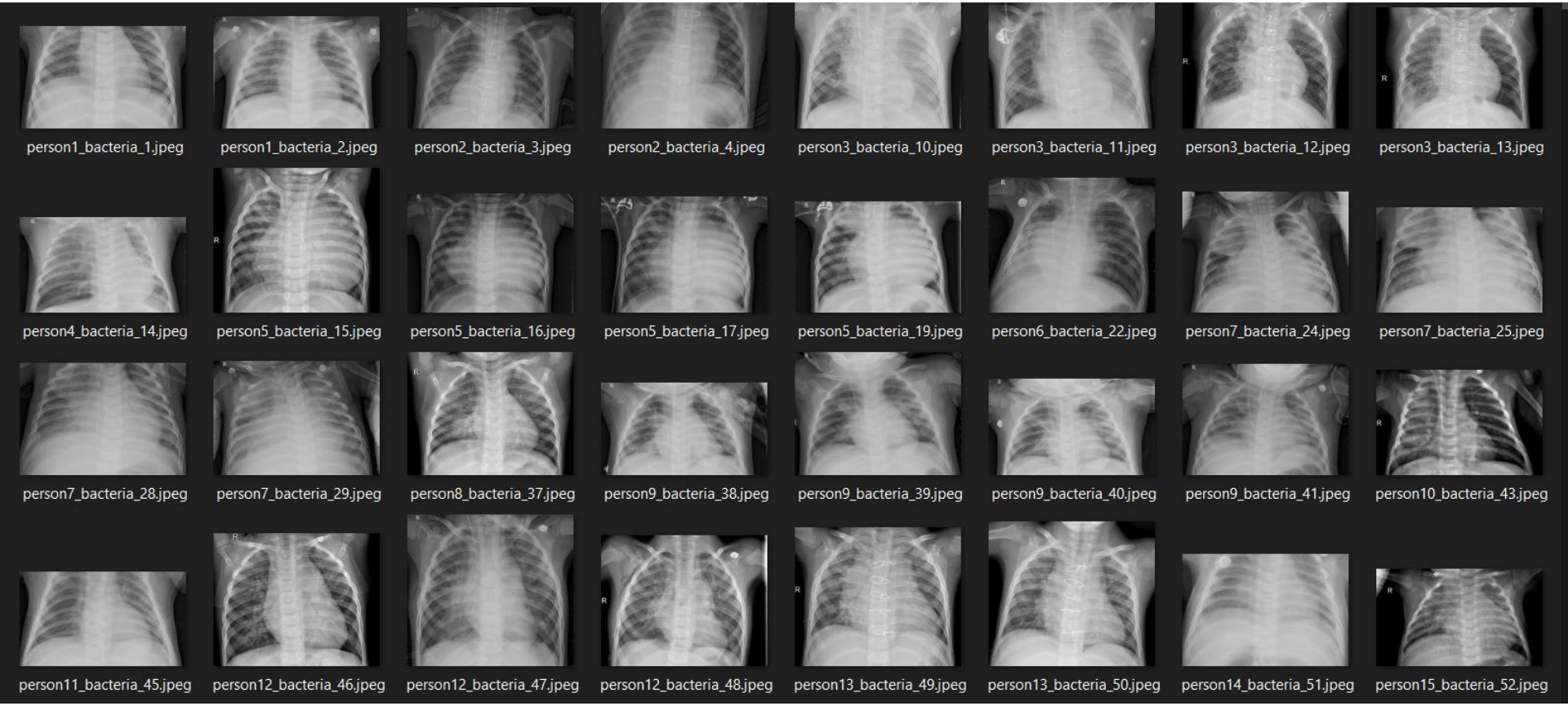


Figure 1: **Left:** Illustration of SGD optimization with a typical learning rate schedule. The model converges to a minimum at the end of training. **Right:** Illustration of Snapshot Ensembling. The model undergoes several learning rate annealing cycles, converging to and escaping from multiple local minima. We take a snapshot at each minimum for test-time ensembling.

- Model can visit multiple local optima

Data augmentation

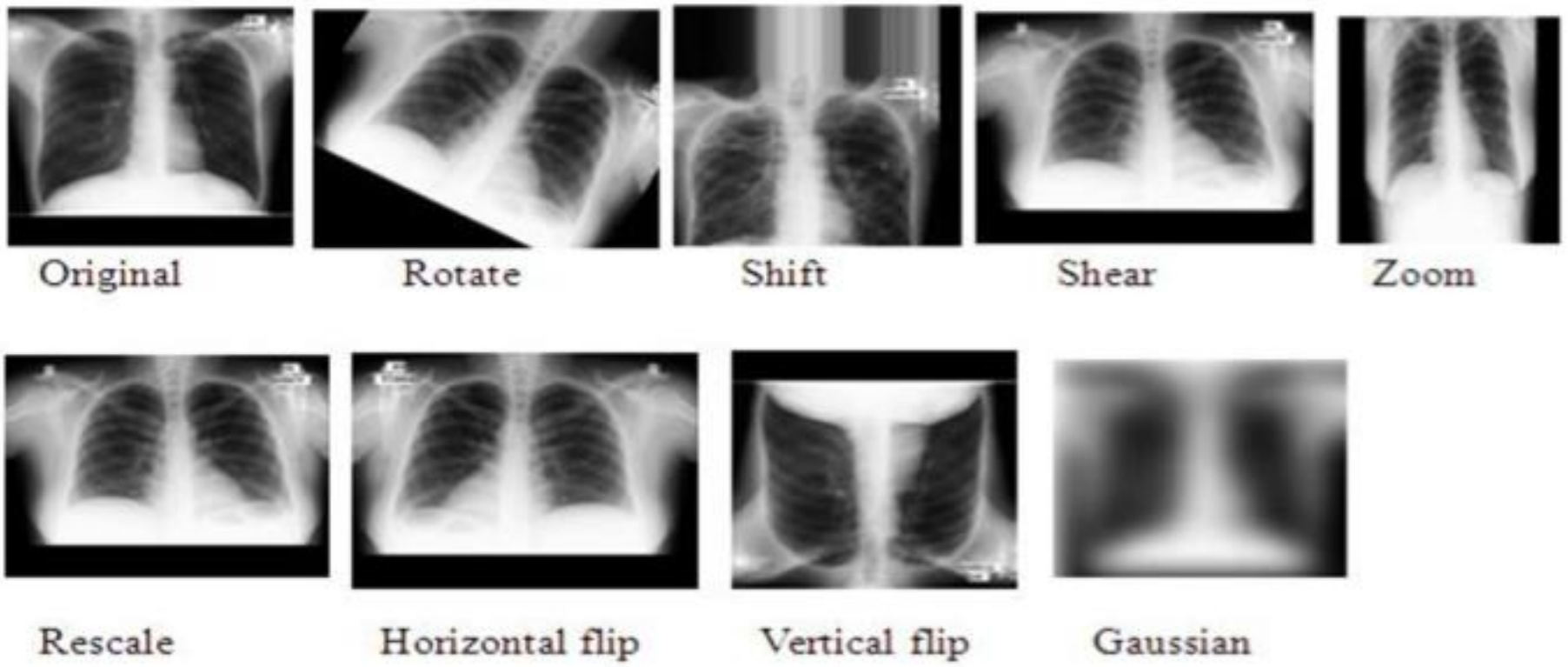
Real data have variations



Source: analyticsvidhya.com

- ANN should be robust to data noises and variations

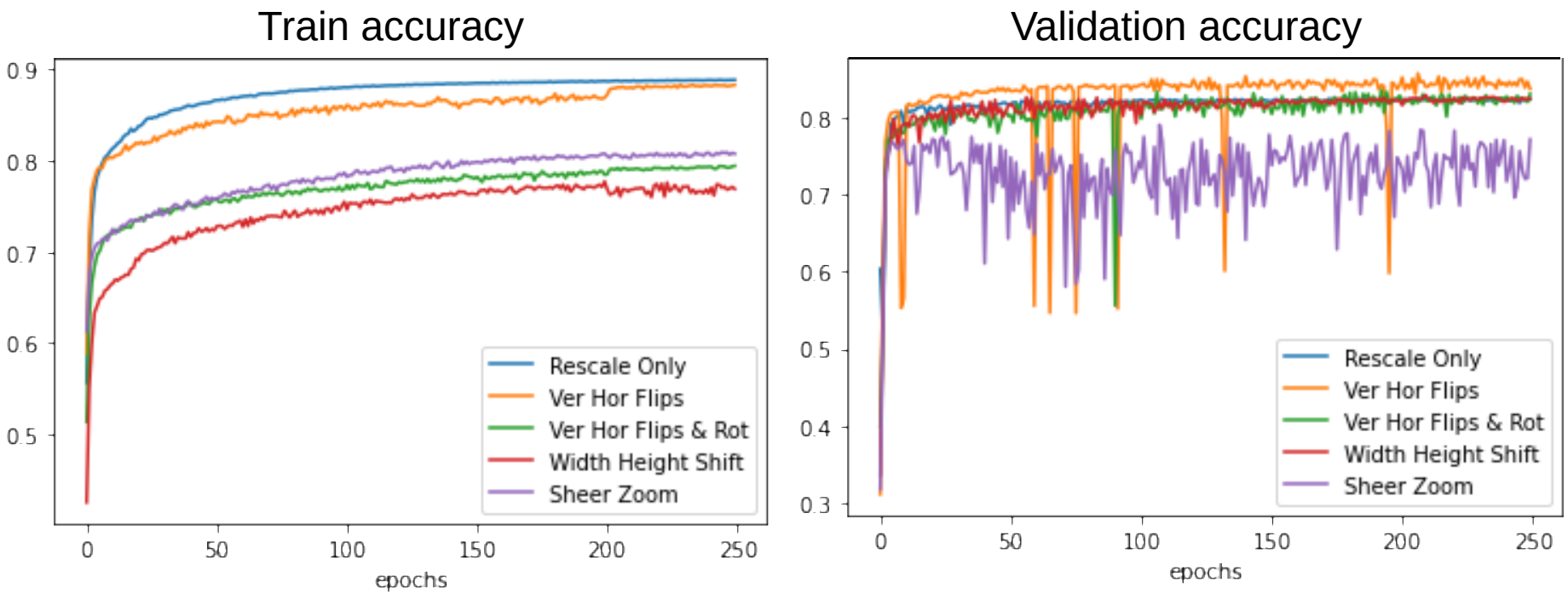
Data augmentation



Rama et al. TIPCV (2019)

- Randomly perform on each batch of training data
- Help model learn features that are generalized

Impact of data augmentation

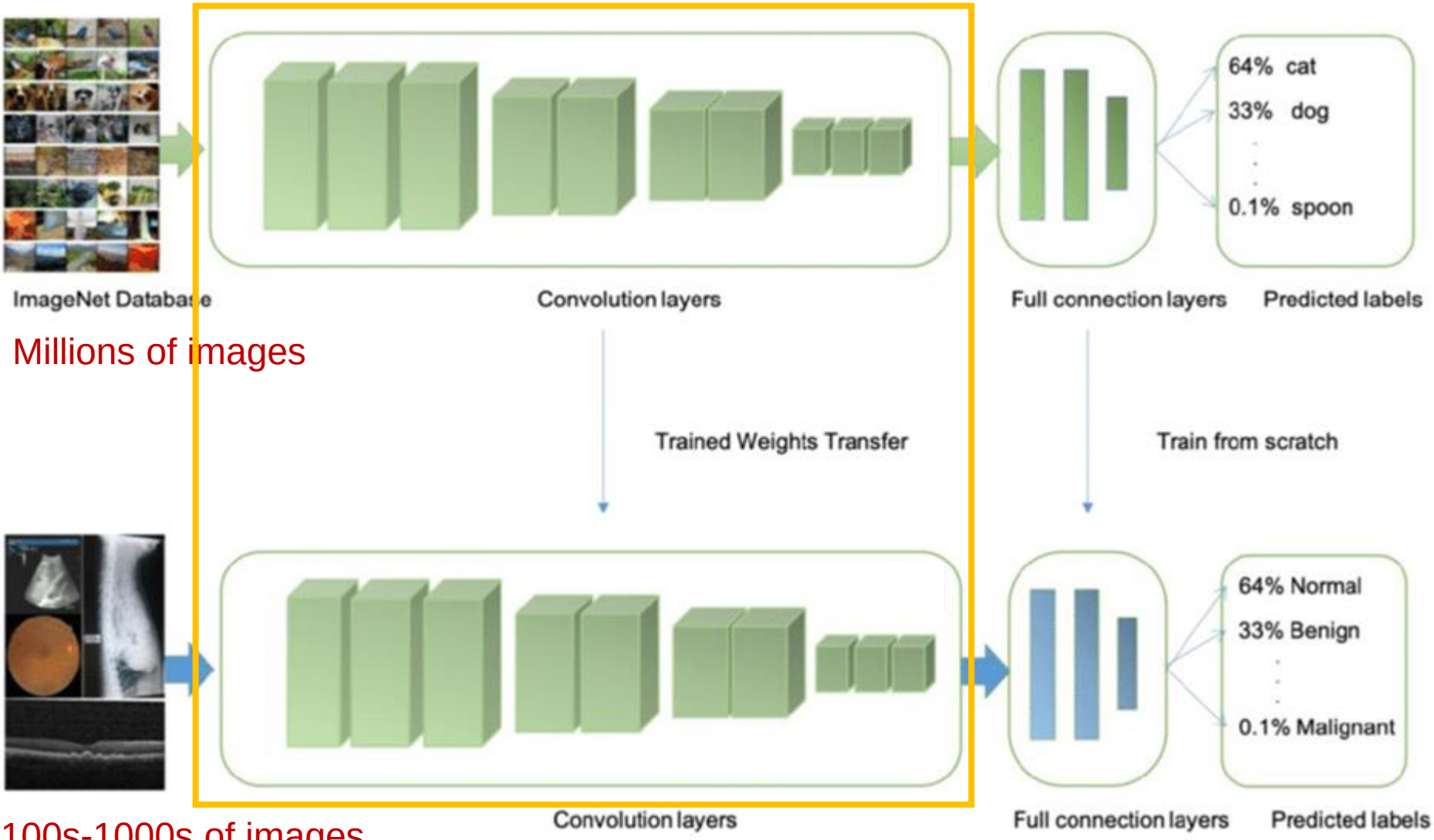


Source: towardsdatascience.com

- Too much augmentation → low training accuracy
- Proper augmentation → improved validation
  - Vertical + horizontal flip

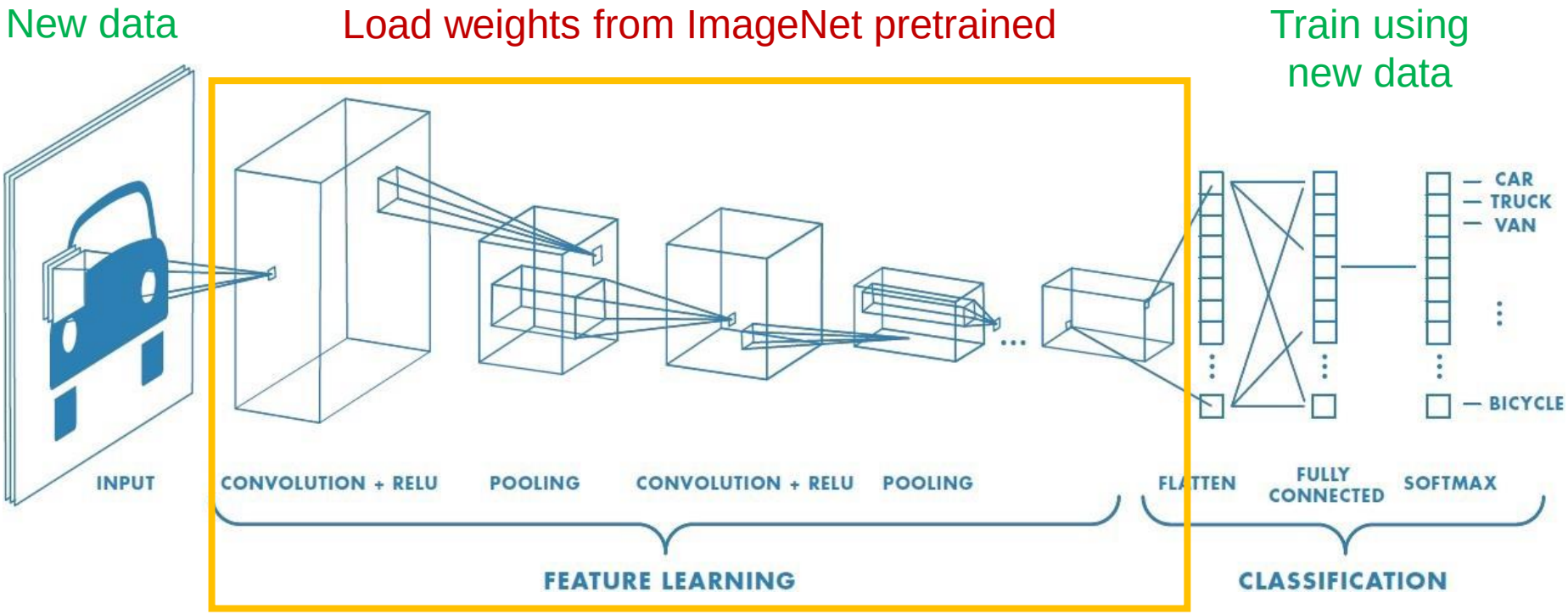
Transfer learning

Transfer learning



Source: Xu et al. Theranostics (2019)

Transfer learning



- Fully connected part is untrained
- Convolution part is pretrained
  - Freeze its weights

Fine-tuning

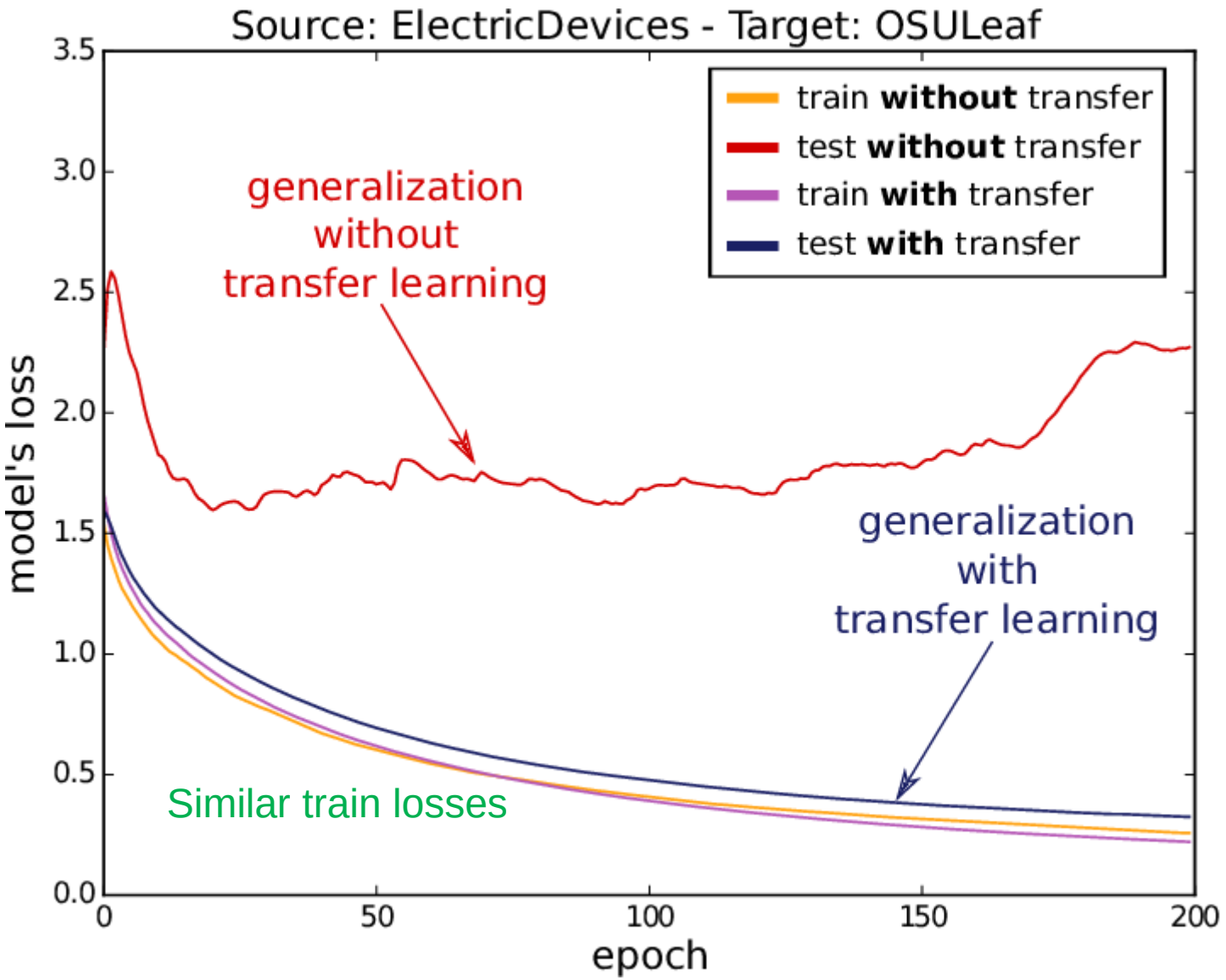
Transfer learning

- Load pretrained model
- Freeze pretrained layers
- Attach to new fully connected layers
- Train model on new dataset

Fine-tuning

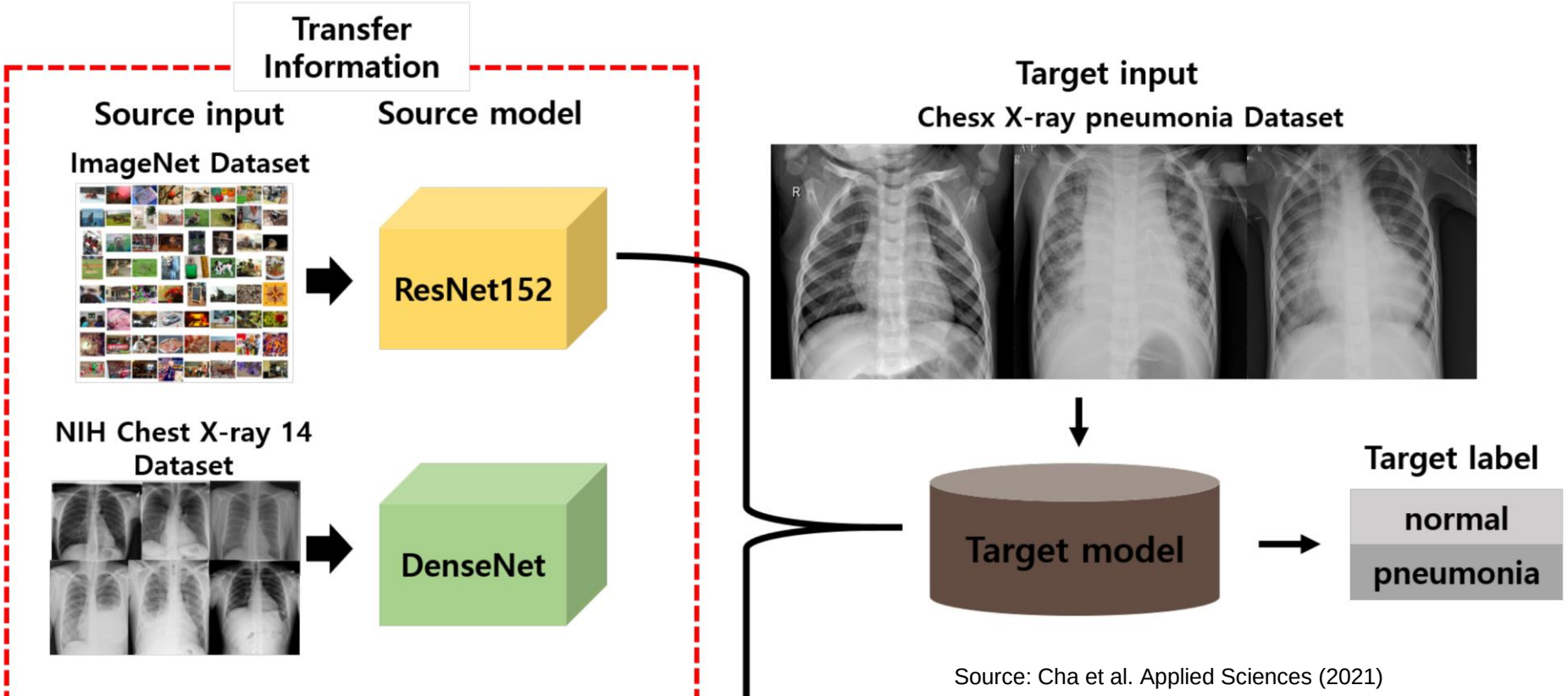
- Unfreeze pretrained layers
- Continue to train the model using new dataset

Impact of transfer learning



Source: Fawaz et al. IEEE Big Data (2018)

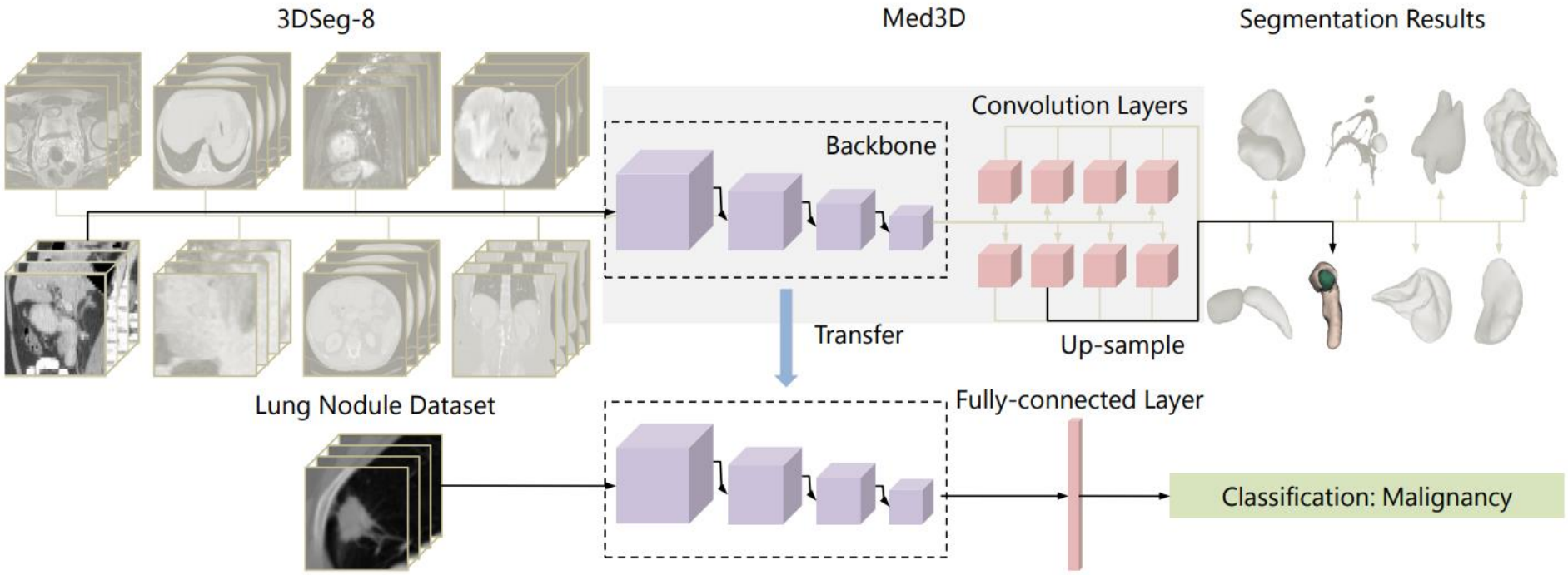
Transfer learning application



Source: Cha et al. Applied Sciences (2021)

- Add new classification
- Adapt to dataset from new institutes

Aggregating datasets with shared model

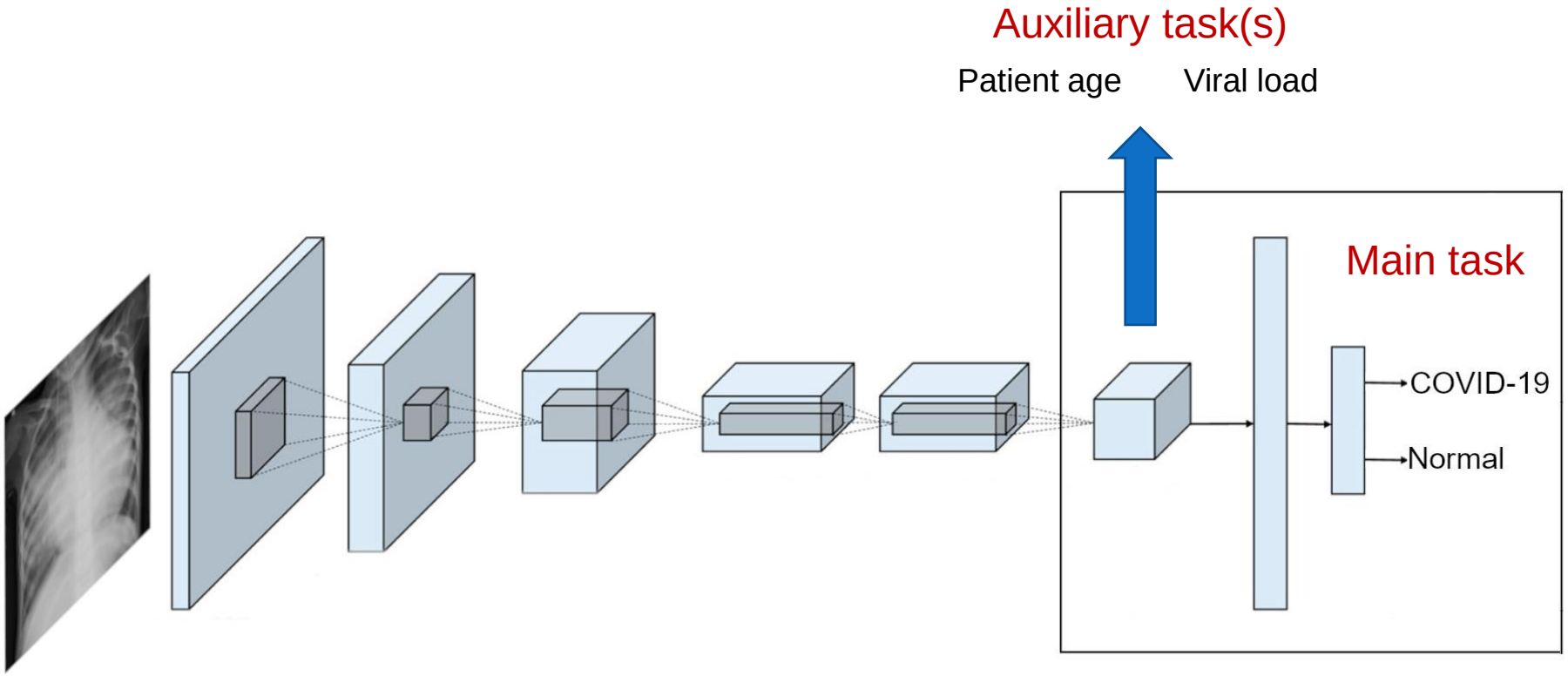


Chen et al. arxiv.org/pdf/1904.00625 (2019)

- Common convolutional layers as encoder for all tasks
- Attach separate output and train using data for each task
- Use ResNet as backbone
- Trained model weights: https://github.com/Tencent/MedicalNet

Auxiliary task

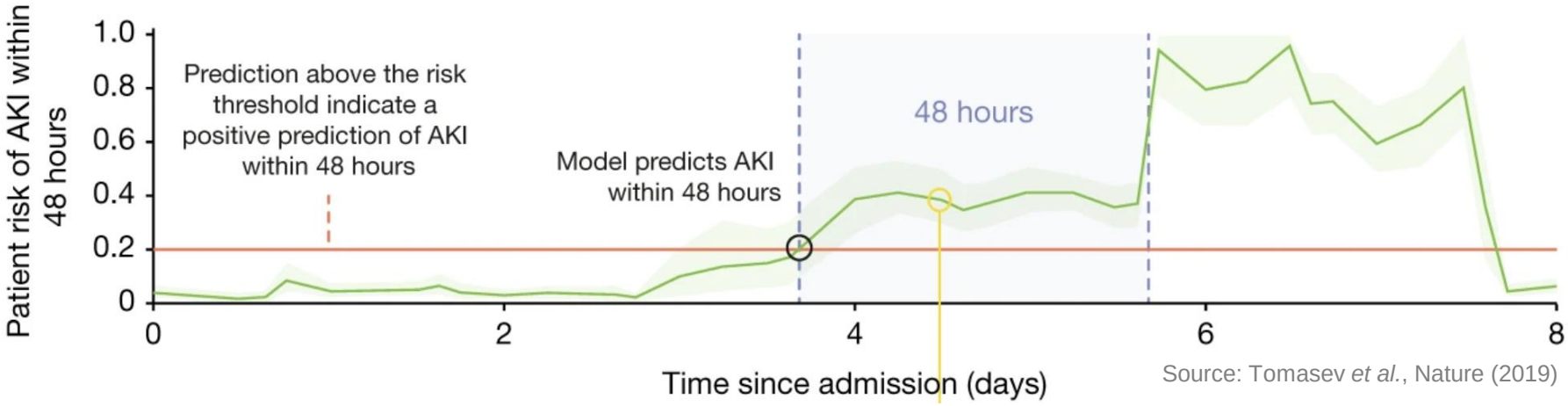
Auxiliary task



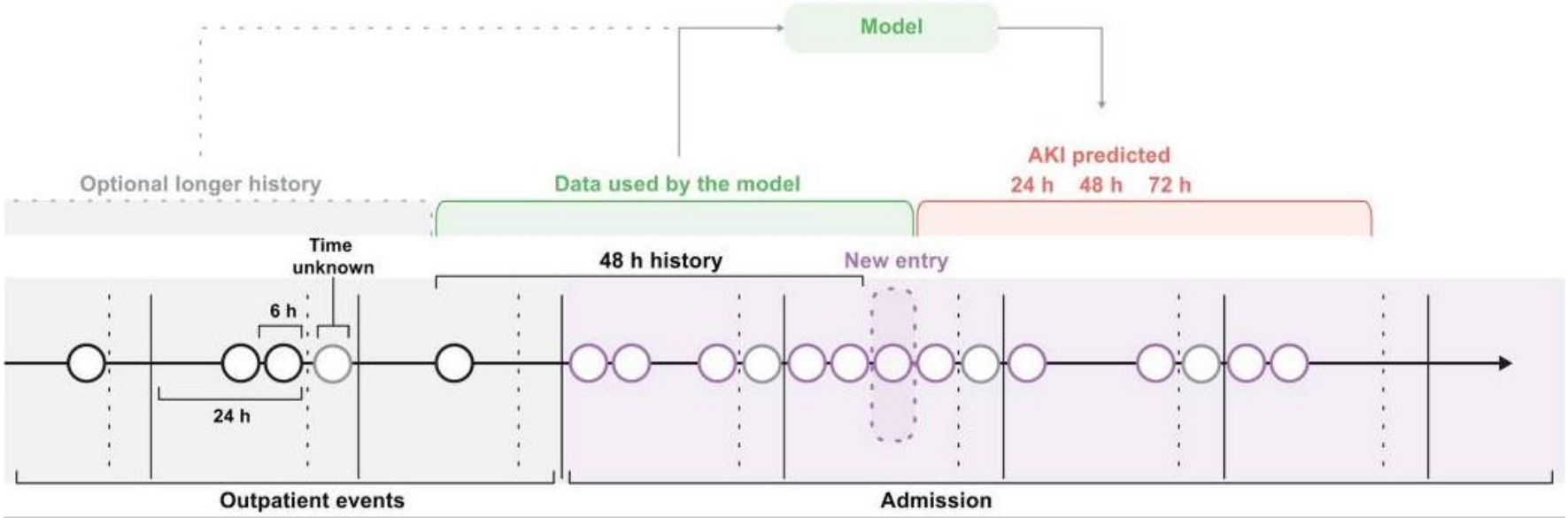
Source: Cortes and Sanchez. IEEE Latin America Transaction (2021)

- Good representation should capture multiple characteristics of the data
- Auxiliary task(s) guides the early layers toward good representation

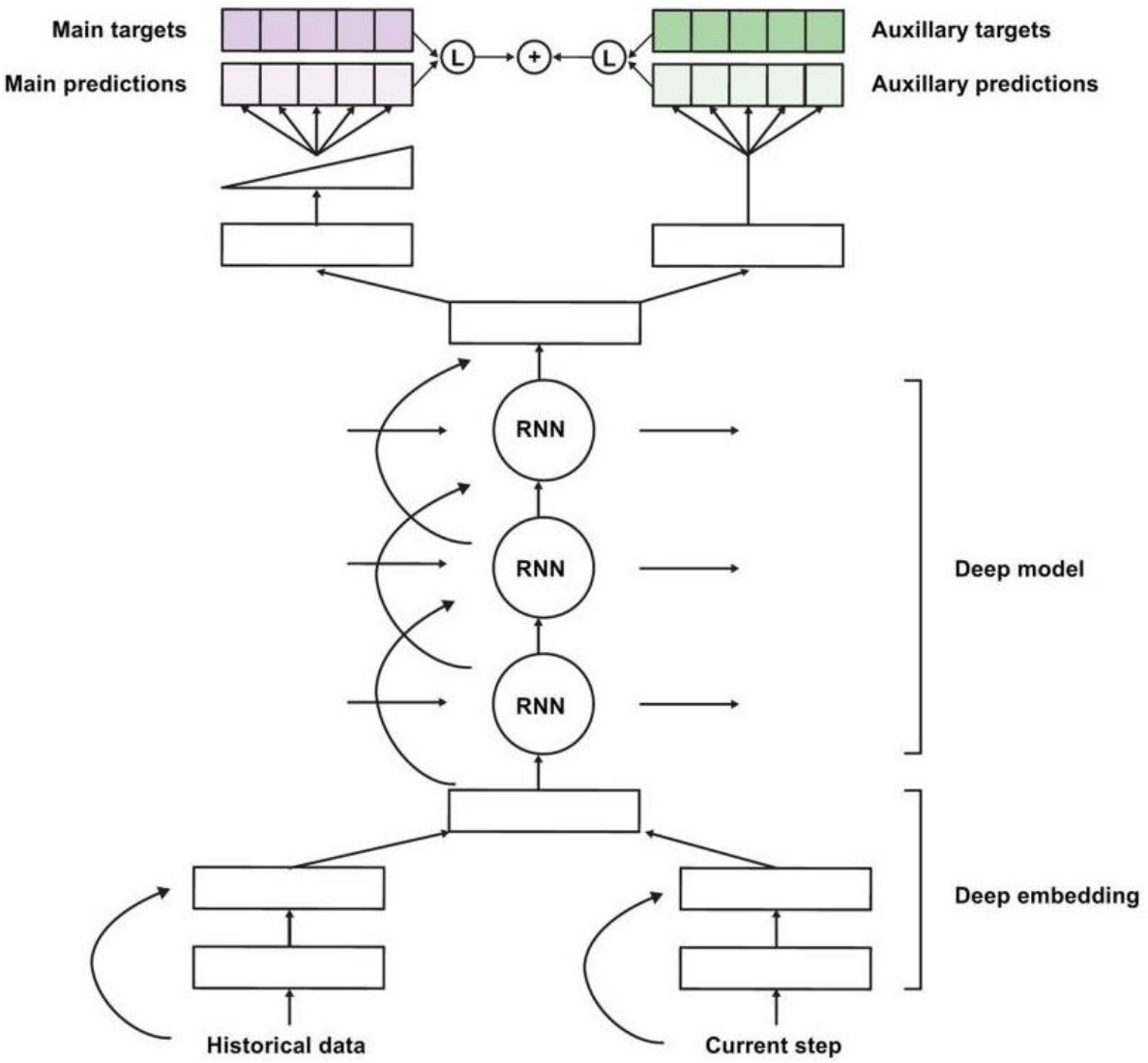
Acute kidney injury prediction



Source: Tomasev *et al.*, Nature (2019)



Auxiliary tasks



Source: Tomasev *et al.*, Nature (2019)

Main task: Predict occurrence of AKI within the next 48 hours

Auxiliary tasks: Predict maximum values of 7 laboratory features over the next 48 hours

Addition of auxiliary task improve AUC by 3%