JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Representation learning

diiP Summer School, June 11 2024

Assistant Professor Jenni Raitoharju
University of Jyväskylä
Jyväskylä, Finland

What is representation learning?

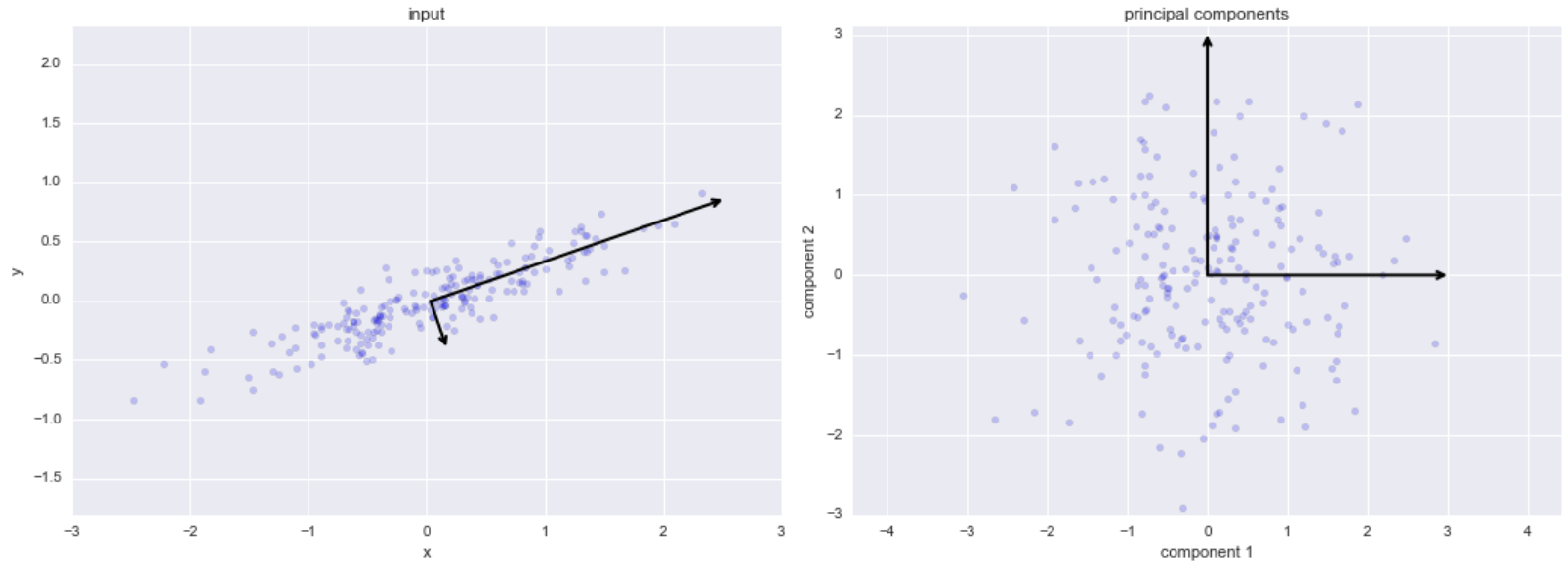
"The **success of machine learning** algorithms generally depends on **data representation**… Although specific domain knowledge can be used to help design representations, **learning with generic priors** can also be used, and the quest for AI is motivating the design of more powerful **representation learning** algorithms implementing such priors." Bengio et al., 2013

"**Representation learning** involves the detection, **extraction**, encoding, and decoding **of features** from raw data, which can then be used in learning tasks. Its objective is to abstract **features that best represent data**." Baghaei et al., 2023

"**Representation learning** is a subset of machine learning where a system learns to **automatically discover the representations** needed for feature detection or classification from raw data", Google Copilot 2024

PCA is a simple unsupervised representation learning approach

- Principal Component Analysis (PCA) is a statistical technique for dimensionality reduction
- It transforms the data into a representation that captures the largest variations in the data



PCA is a simple unsupervised representation learning approach

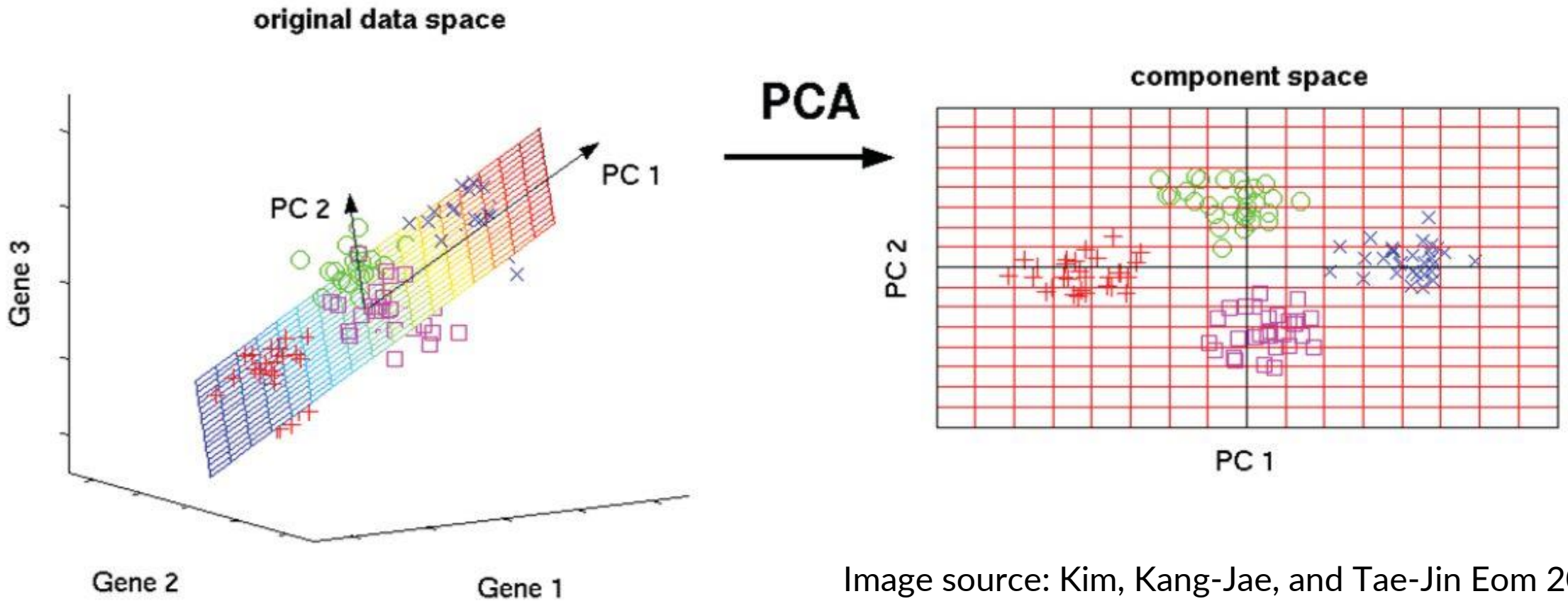


Image source: Kim, Kang-Jae, and Tae-Jin Eom 2016

Benefits of PCA preprocessing

- **Dimensionality reduction:** PCA reduces the number of dimensions in the training dataset, which can help to alleviate issues like the curse of dimensionality and improve the efficiency of machine learning
- **Noise reduction:** By capturing the directions that maximize variance, PCA helps to de-noise the data, emphasizing the signal over the noise
- **Data visualization:** PCA can simplify the complexity of high-dimensional data, making it easier to visualize and understand
- **Improved performance:** With fewer dimensions, algorithms can run faster and often with better results, as irrelevant or redundant features may be discarded
- **Feature independence:** PCA transforms correlated variables into a set of uncorrelated variables, which can be beneficial for models that assume feature independence

LDA is a simple supervised representation learning approach

- Linear Discriminant Analysis (LDA) transforms the data in a way that makes items from different classes easier to separate

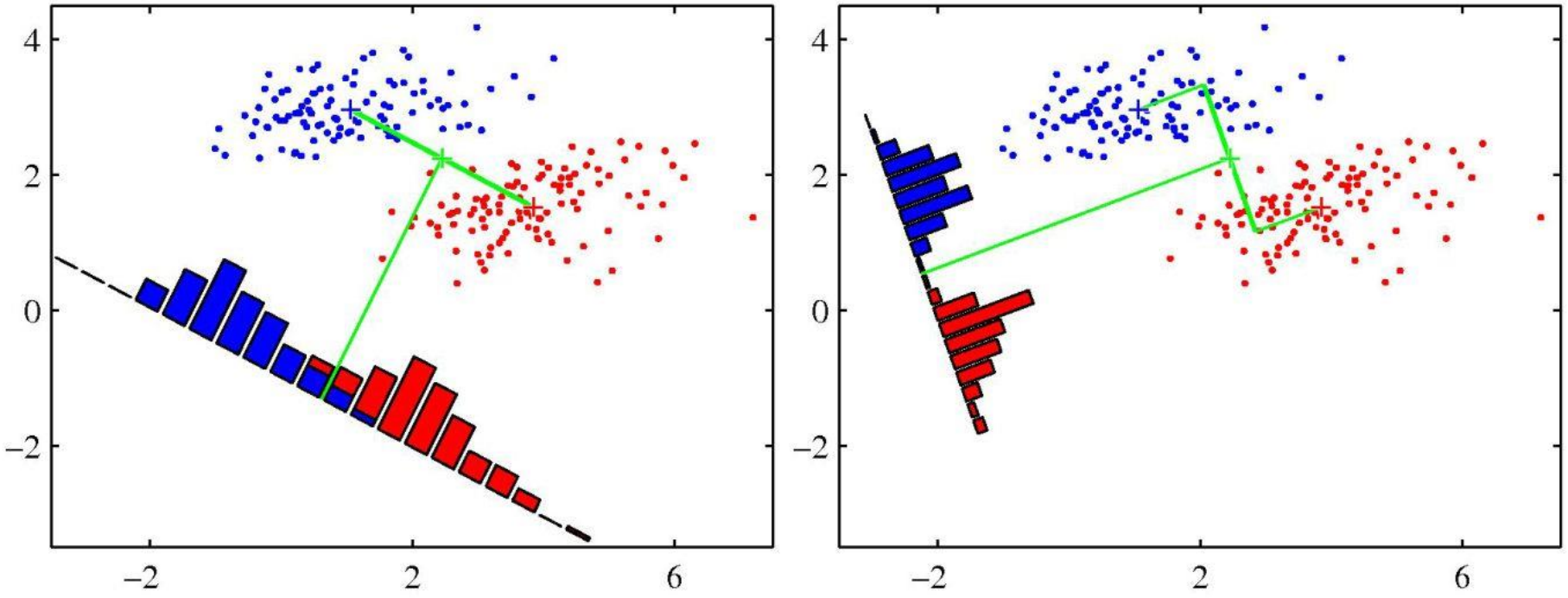


Image source: Iosifidis A., diiP lecture on Supervised learning

More complex inputs require more complex representations

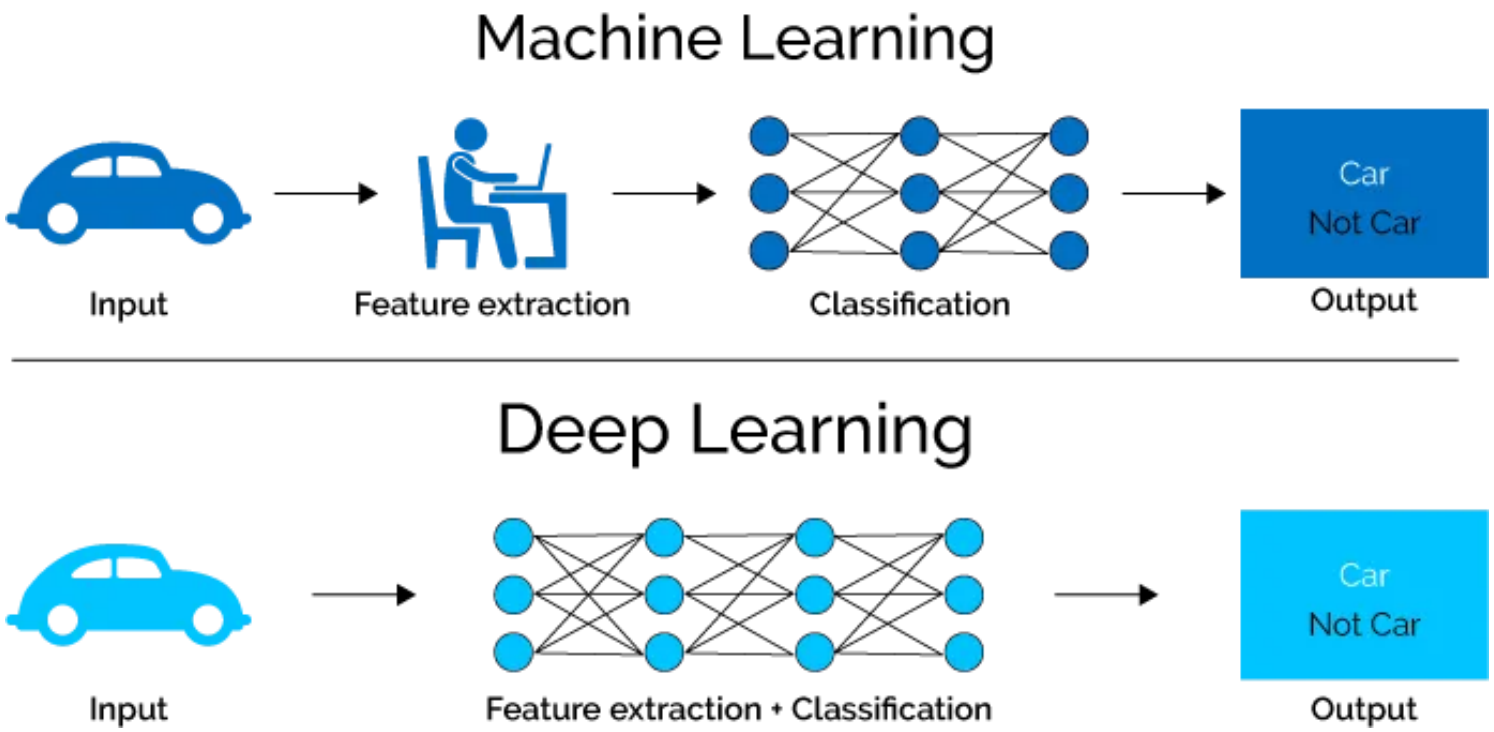


Image source: https://towardsdatascience.com/cnn-application-on-structured-data-automated-feature-extraction-8f2cd28d9a7e

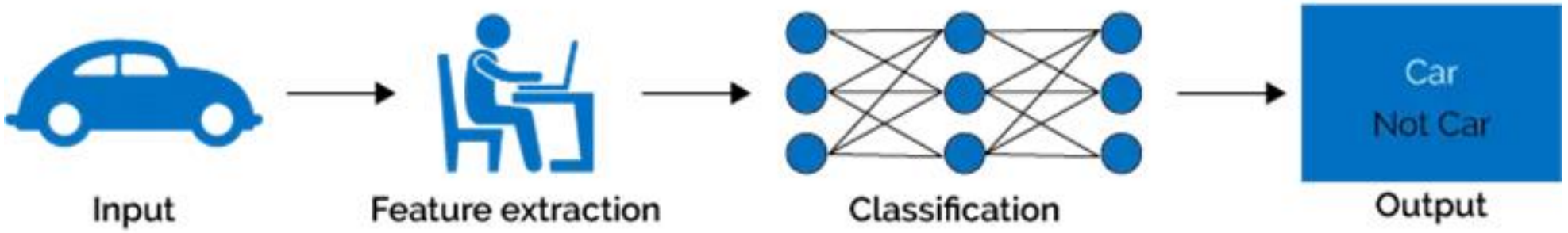
- Representation learning is automated feature extraction

Lecture outline

- Handcrafted feature extraction
- CNNs as representation learners
- Different approaches for representation learning
  - Supervised, unsupervised, semisupervised
- Transfer learning
- Domain adaptation
- Challenges in representation learning

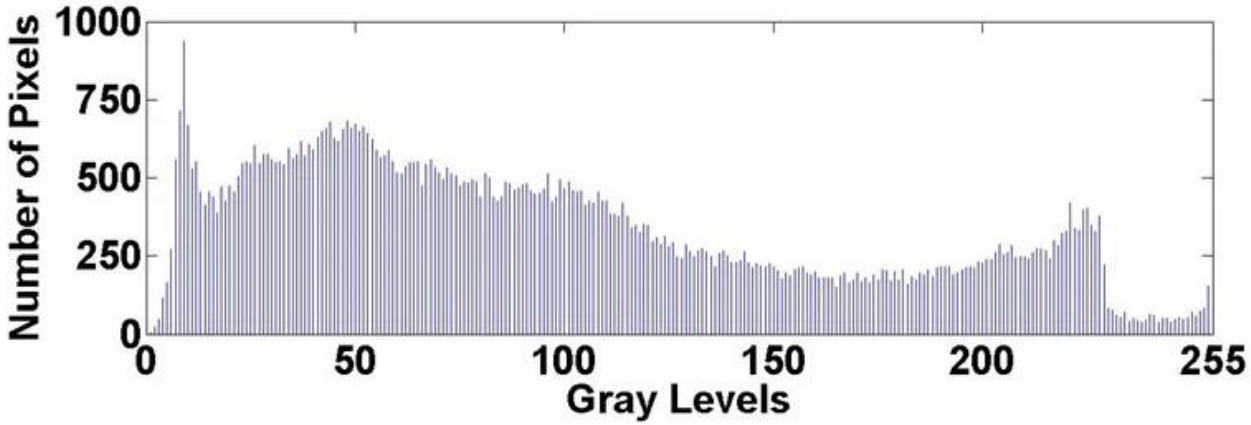
Handcrafted feature extraction

Historical (-2010) approach for machine learning



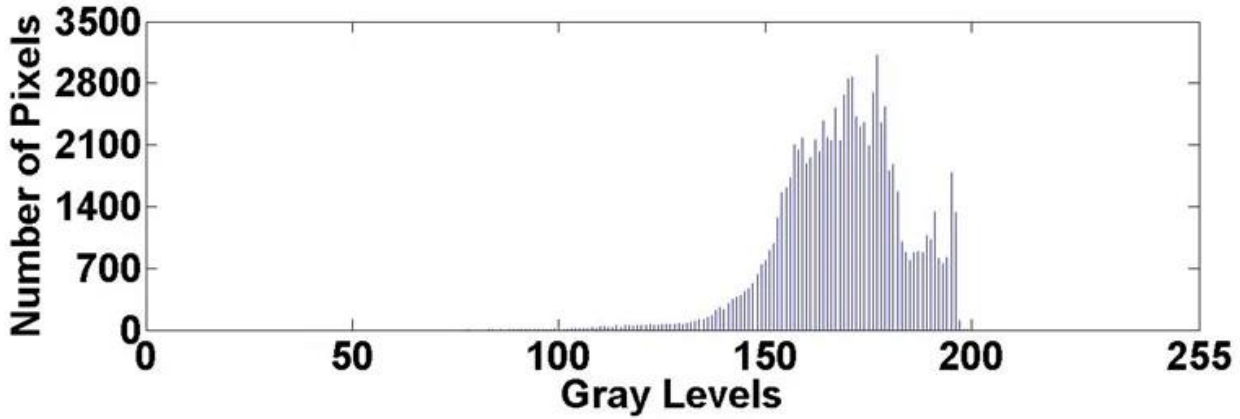
- A lot of research in the past focused on developing better feature extraction methods for different tasks
- Optimal representation (features) depends on the task

Histograms as image representations



(a)

(b)



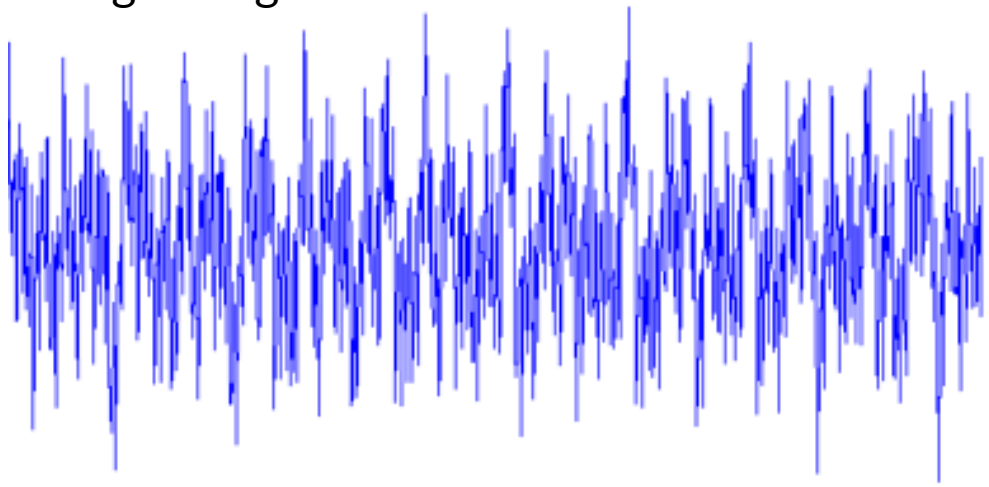
(a)

(b)

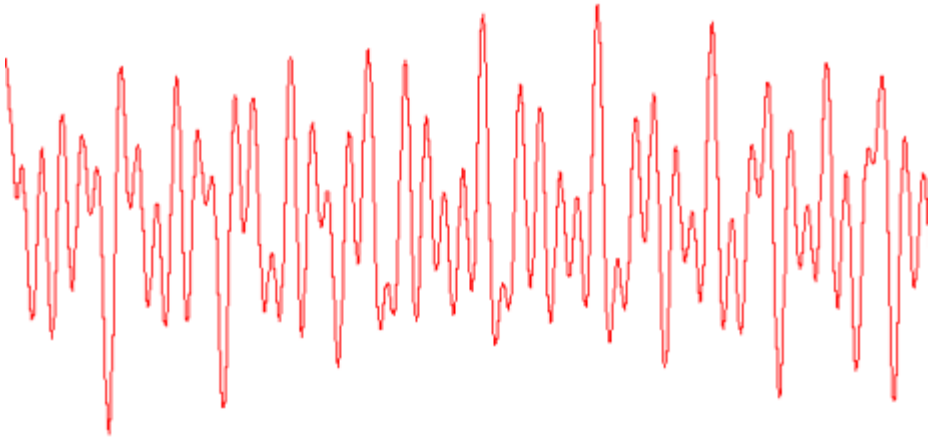
Image source: https://www.allaboutcircuits.com/technical-articles/image-histogram-characteristics-machine-learning-image-processing/

Feature extraction via filtering

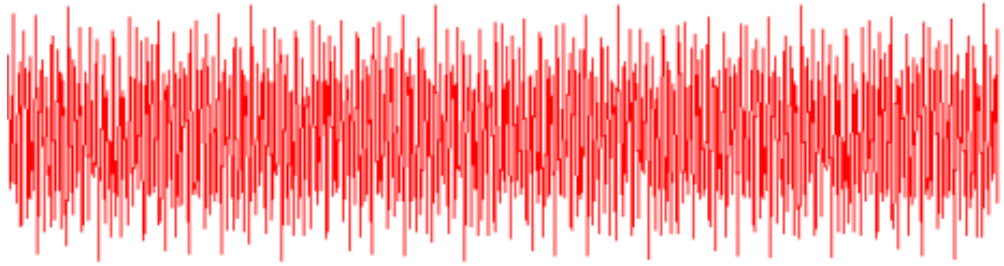
Original signal



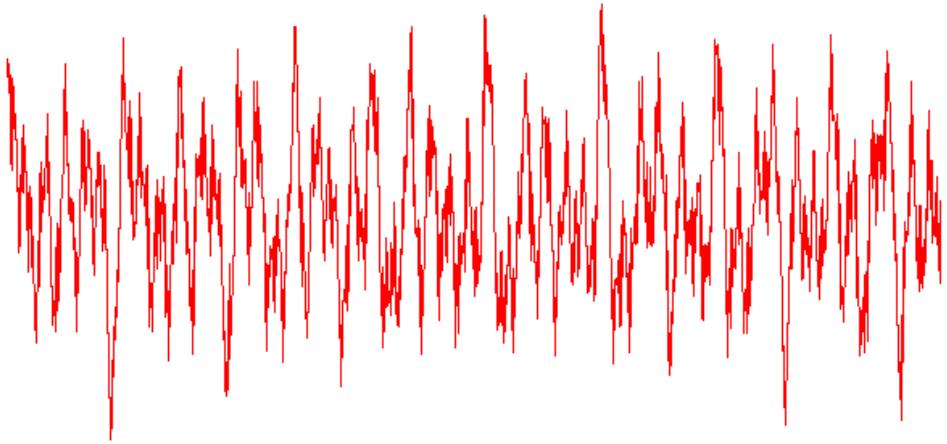
Lowpass filtered signal



Highpass filtered signal



Bandpass filtered signal



Filtering via convolution

- In signal processing, filtering is performed via convolution between the original signal and the filter (another finite/infinite signal)

$$f[n] * g[n] = \sum_{k=-\infty}^{\infty} f[k]\, g[n-k]$$

- We just need to design suitable filters for each task

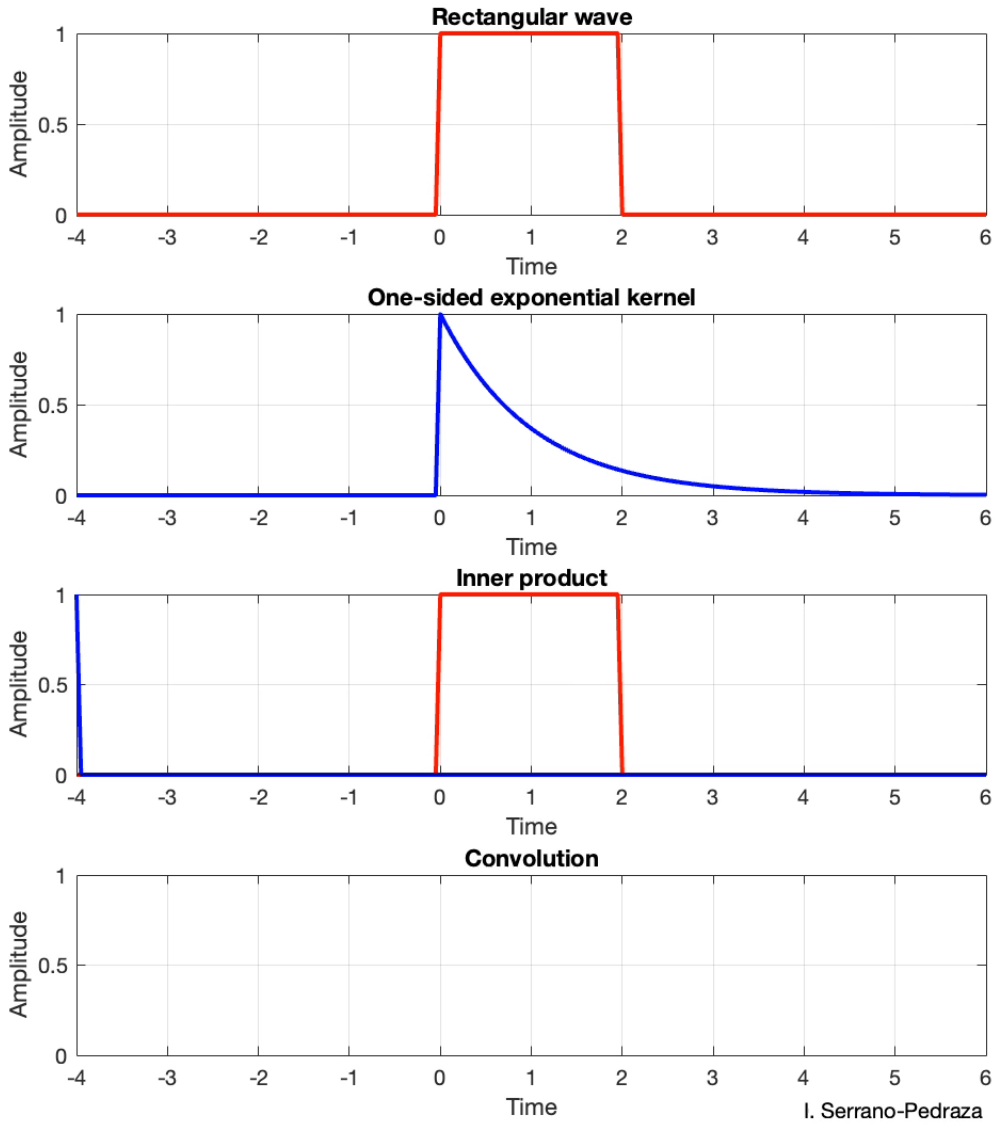
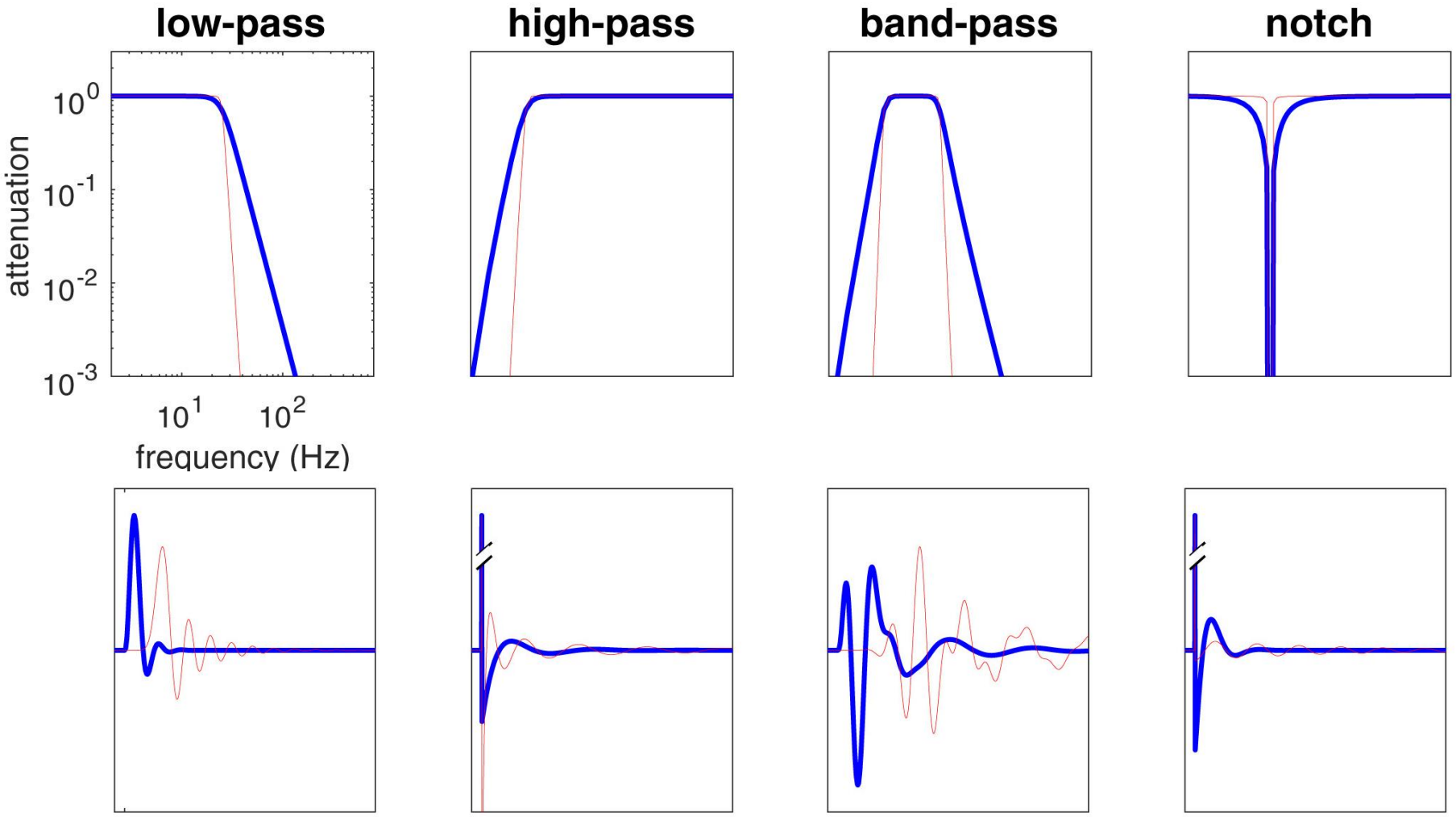


Image source: https://www.sciencedirect.com/science/article/pii/S0896627319301746#fig3

Filters designed for different tasks



Desired frequency change

A filter designed for the task

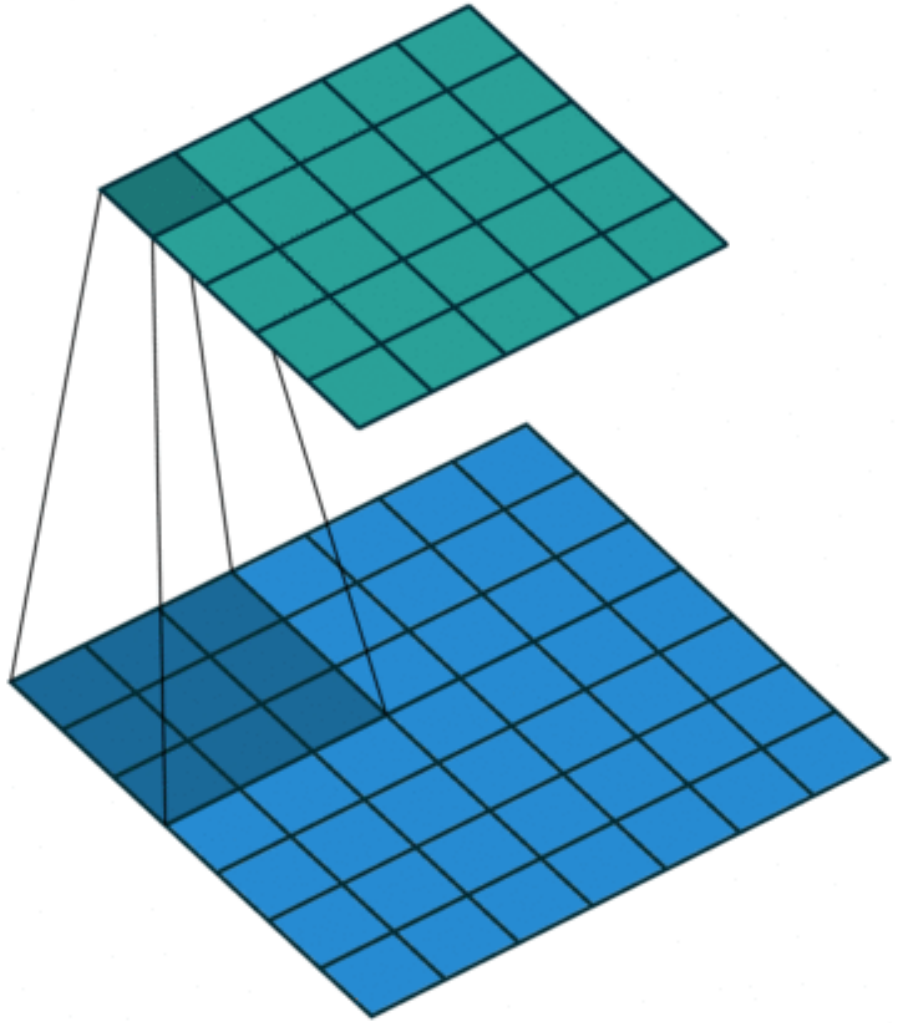
Image source: https://www.ucm.es/serranopedrazalab/other

2D filtering of images

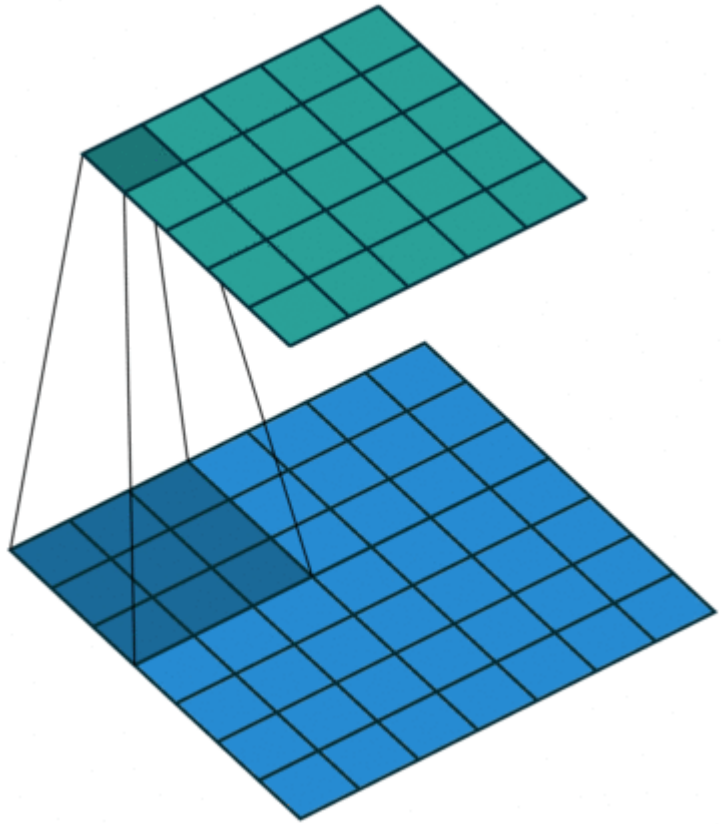
- Images can be similarly filtered using 2D filters and convolution (correlation)
- Image processing filters are usually KxK matrices

$$Y[m,n] = \sum_{i=1}^{K}\sum_{j=1}^{K} K[i,j]\,X[m-1+i, n-1+j]$$

- Designing different filters for different image analysis tasks used to be a common task ~15 years ago



2D filter examples



*

Average filter

$$\begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix}$$

*

Sobel filter

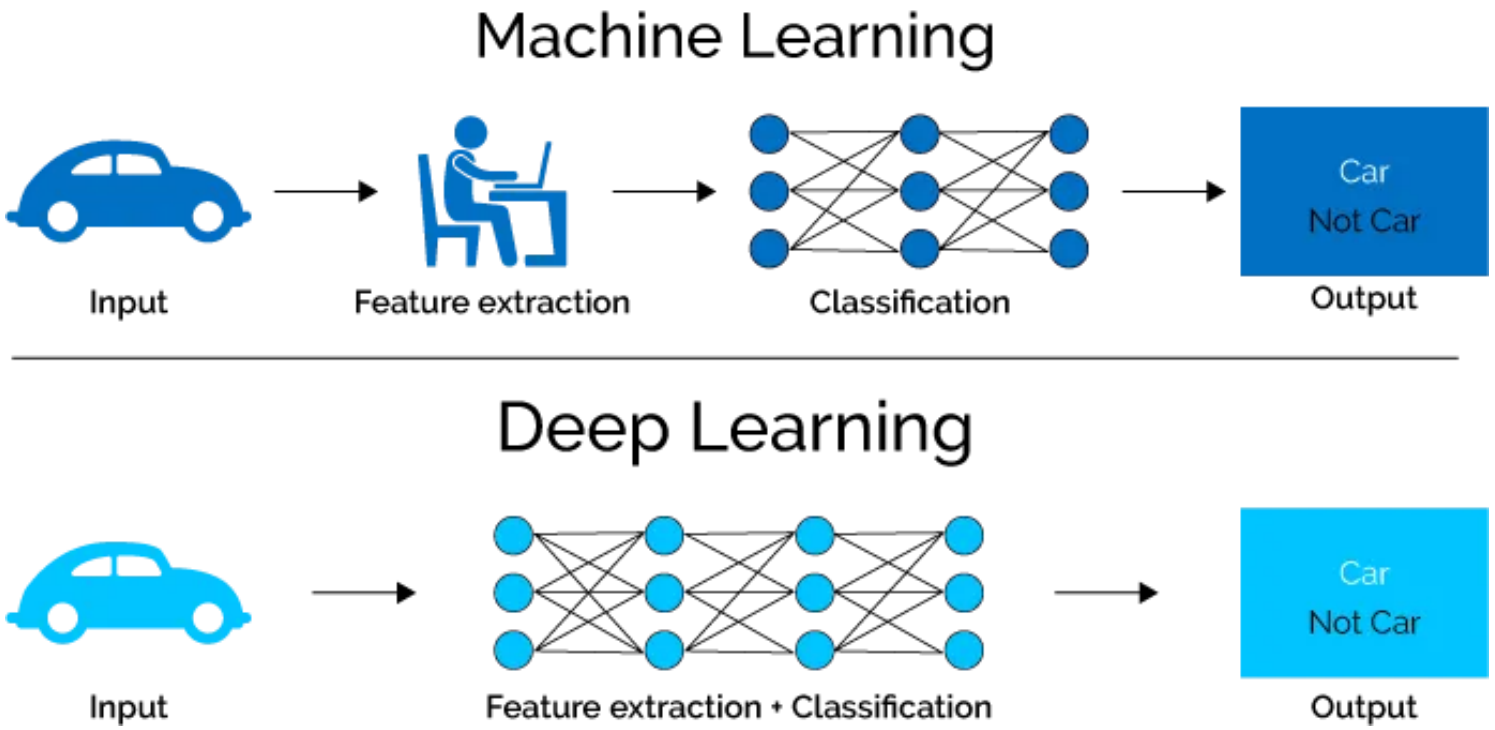
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Image source: Raitoharju, 2022

CNNs as repsesentation learners

CNNs as representation learners

- Convolutional Neural Networks (CNNs) can take images as inputs
- They **learn to perform optimal feature extraction**
- Developments in GPU computation enabled CNN breakthrough as it became possible to train with very large training sets



CNNs as representation learners

- Each convolutional layer has multiple parallel **filters**
- Each filter has multiple **channels** (kernels) to handle each channel in the layer input
- Each filter produces one **feature map** (layer)
- **Activation** (activation map) is the output of applying the activation function (e.g., ReLU) to the feature map

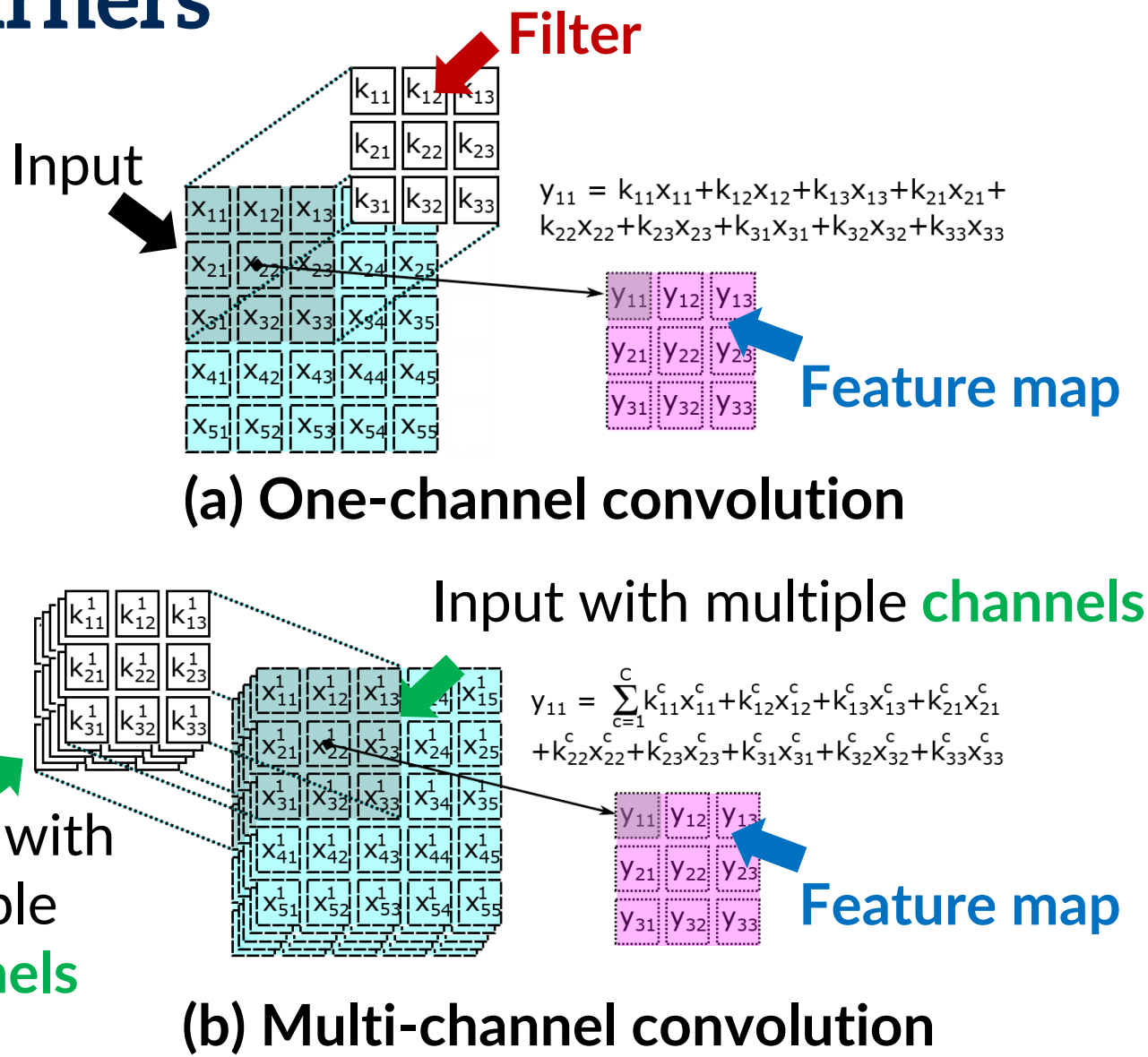
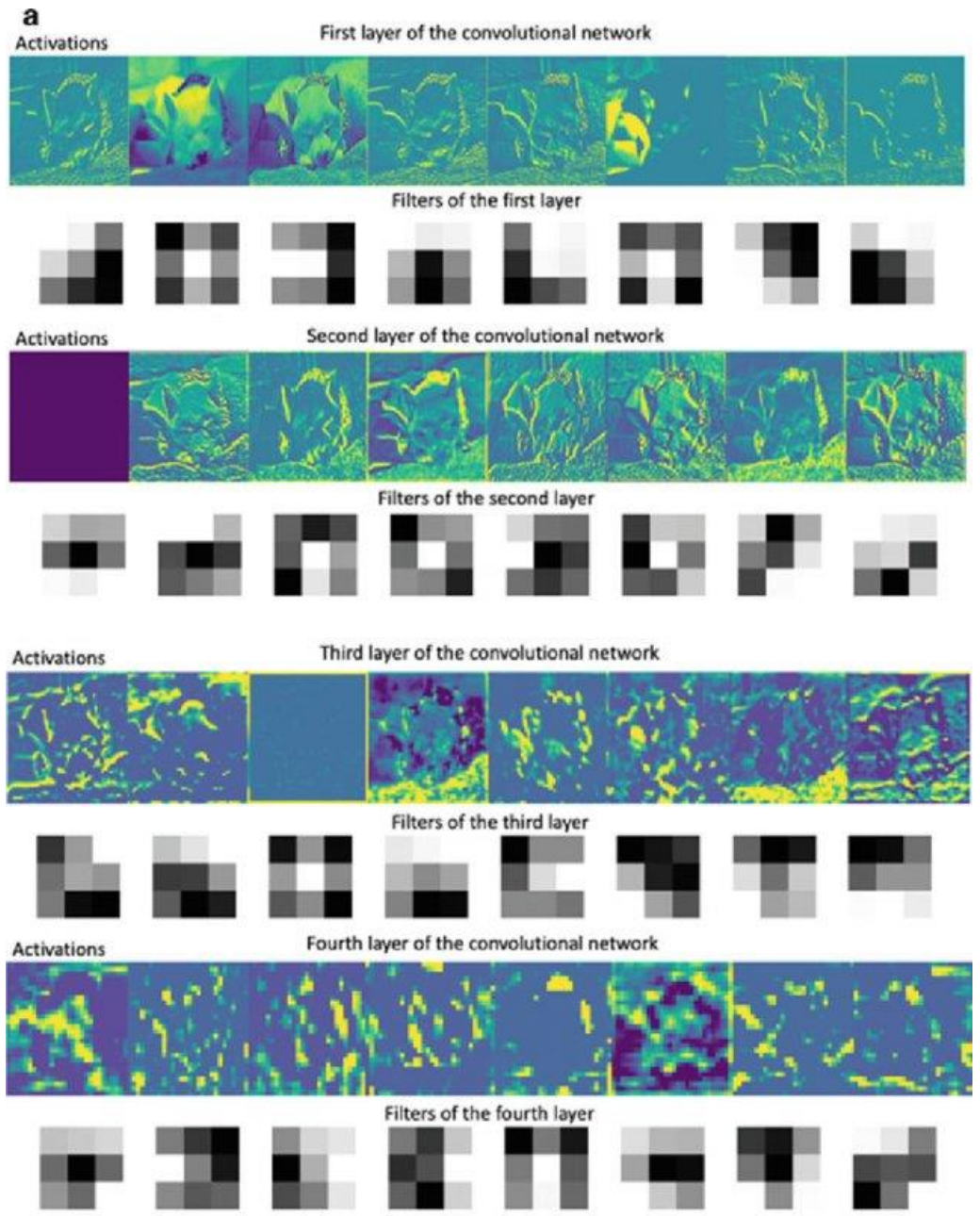


Image source: Raitoharju, 2022

Can we look at what a CNN is learning?

- **Filter** weights
- **Activation** maps (representations) for each input image
- Part of the input image that leads to the strongest activation



Visualization of the first eight activations and filters of layers 1, 2, 3 and 4 in the VGG16 network trained with ImageNet.
Image source: Ng and Feng, 2020

AlexNet architecture

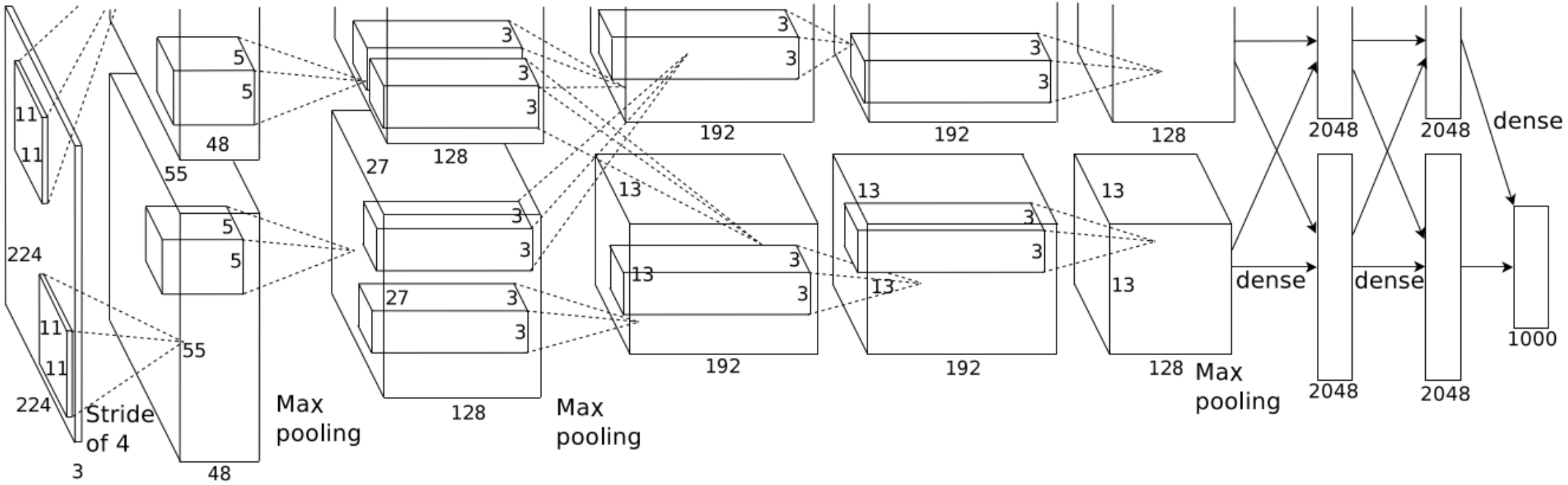
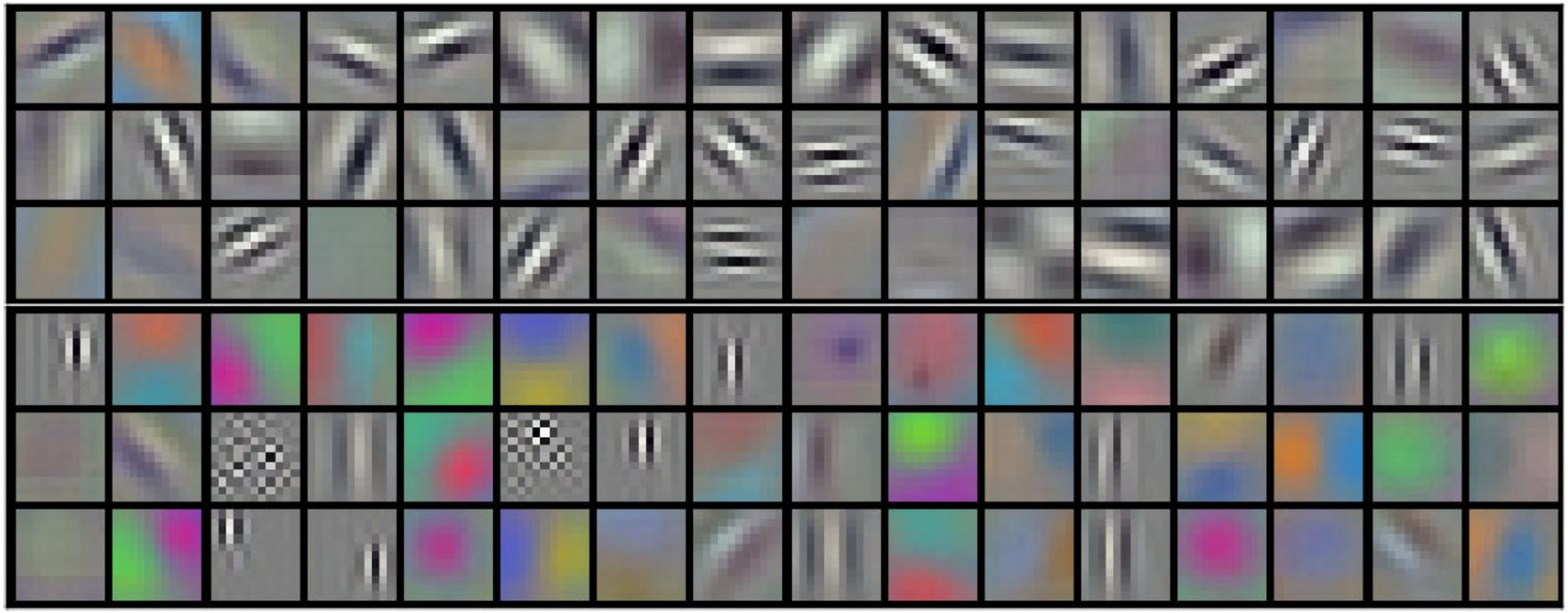


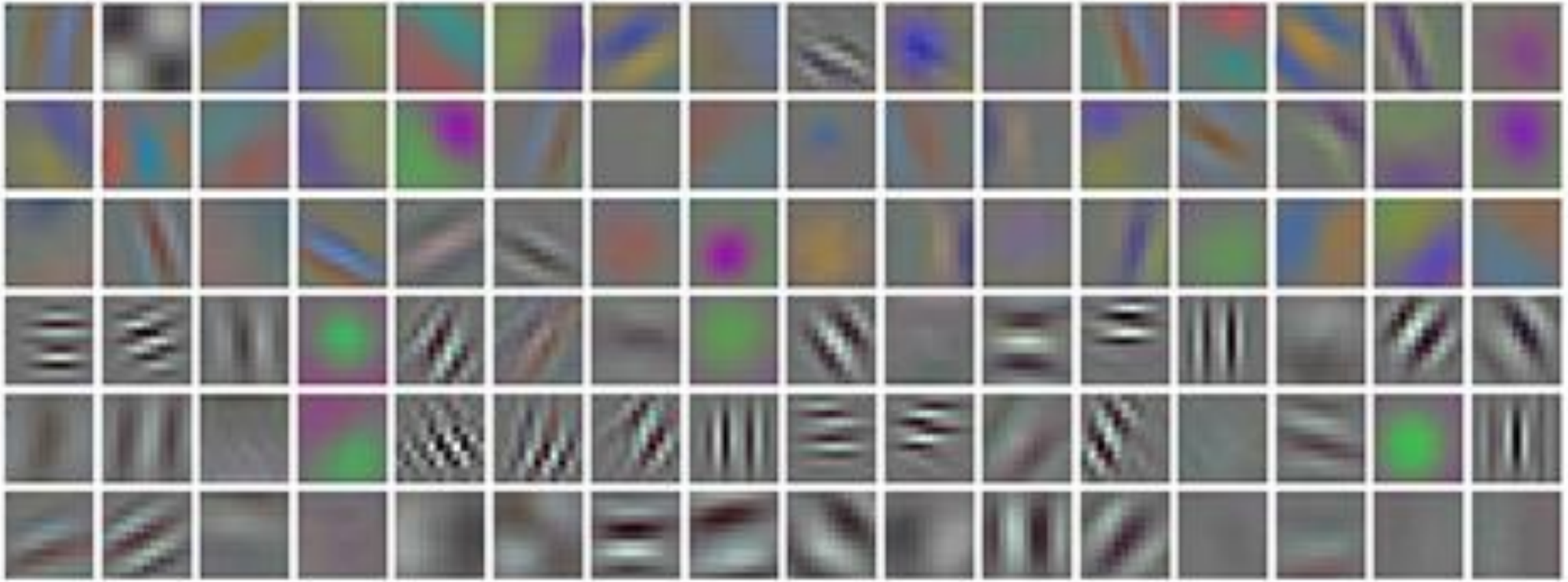
Image source: Krizhevsky et al. 2012

AlexNet filter weights for ImageNet



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images
Image source: Krizhevsky et al. 2012

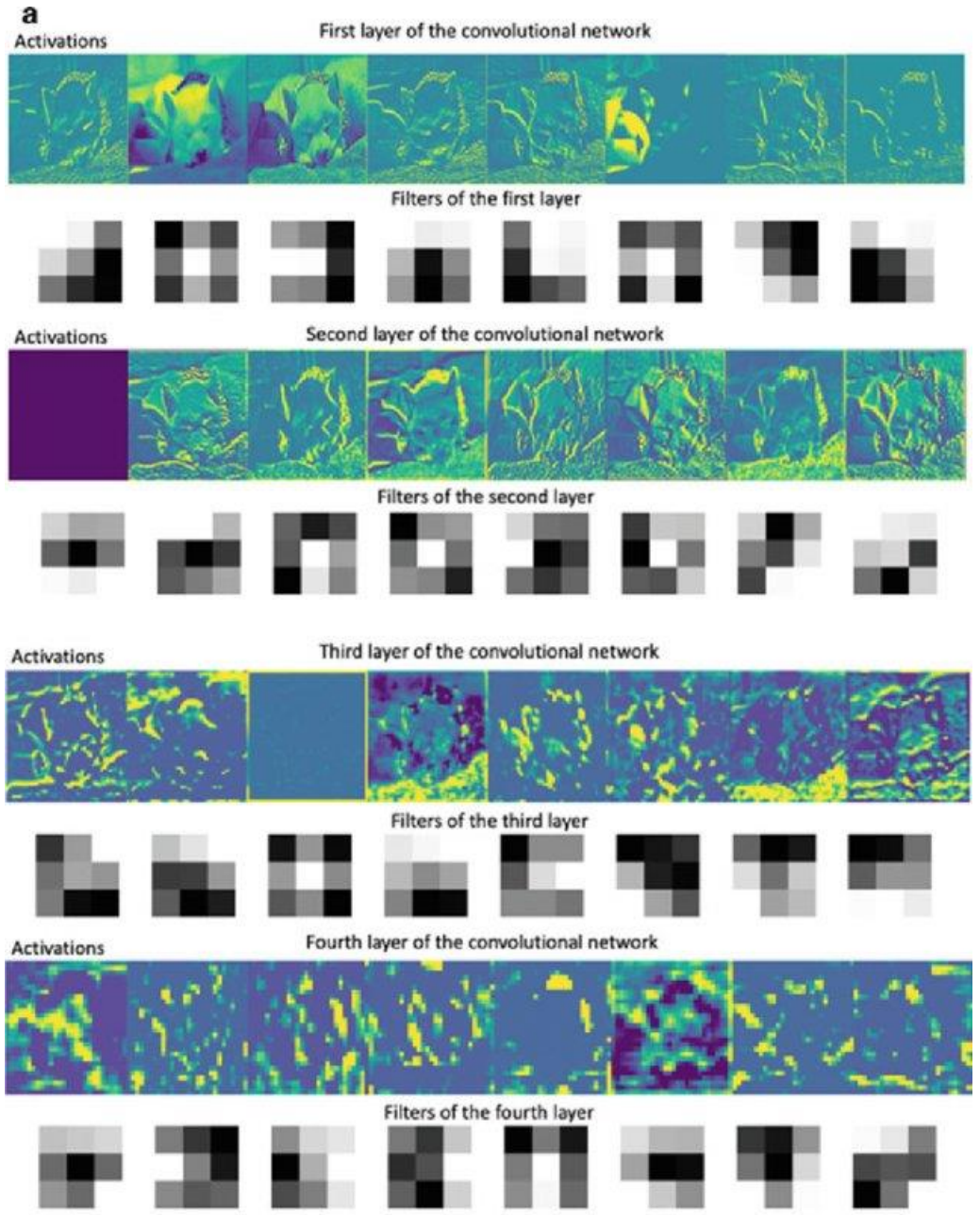
AlexNet filter weights for artwork detection



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images
Image source: Brachmann et al. 2017

VGG16 filters and activations for ImageNet

- **Filter** weights for different layers can be quite similar
- After multiple convolution layes, the **activations** get more and more difficult for humans to interpret



Visualization of the first eight activations and filters of layers 1, 2, 3 and 4 in the VGG16 network trained with ImageNet.
Image source: Ng and Feng, 2020

DenseNet architectures

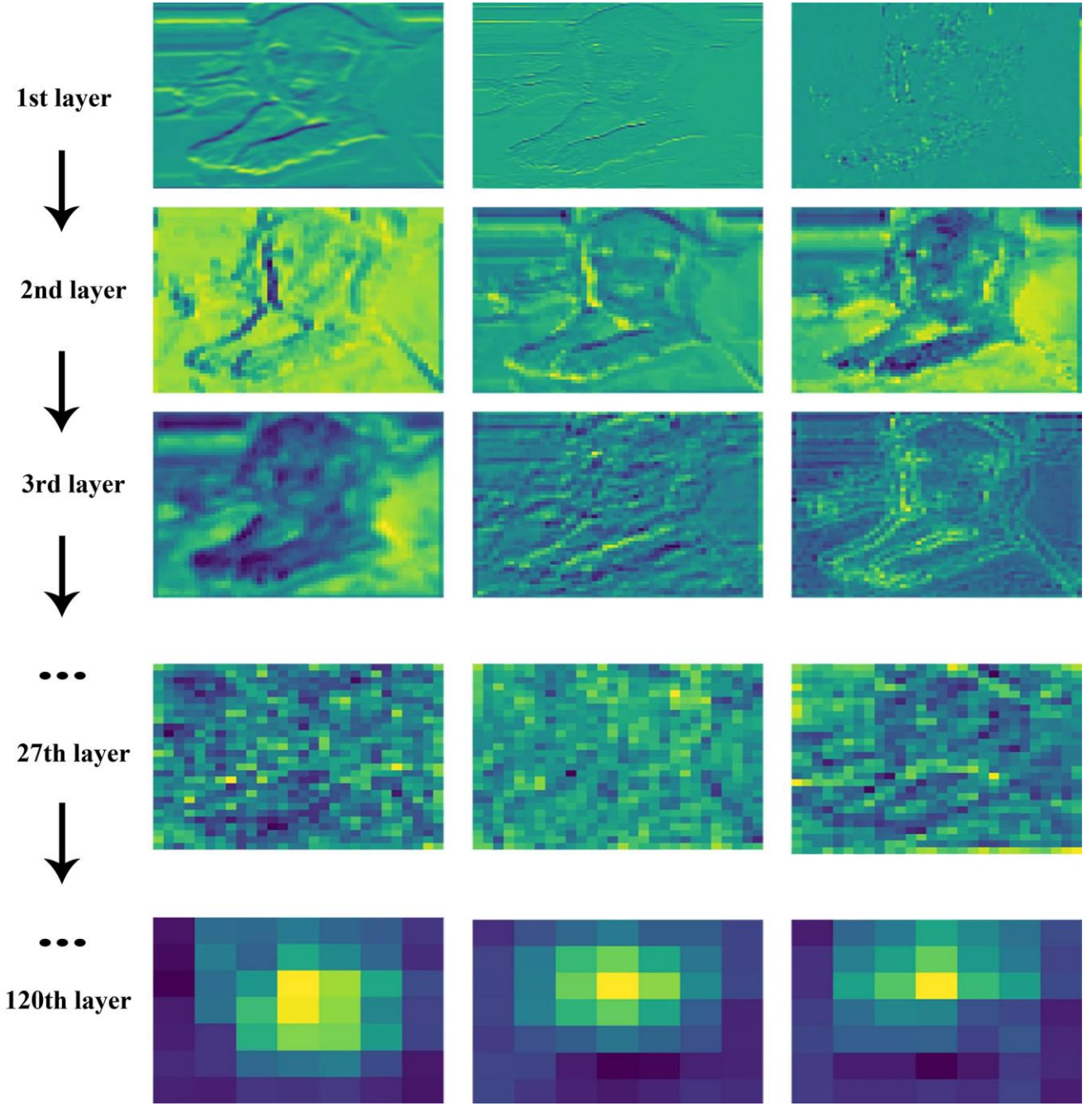
Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1: DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

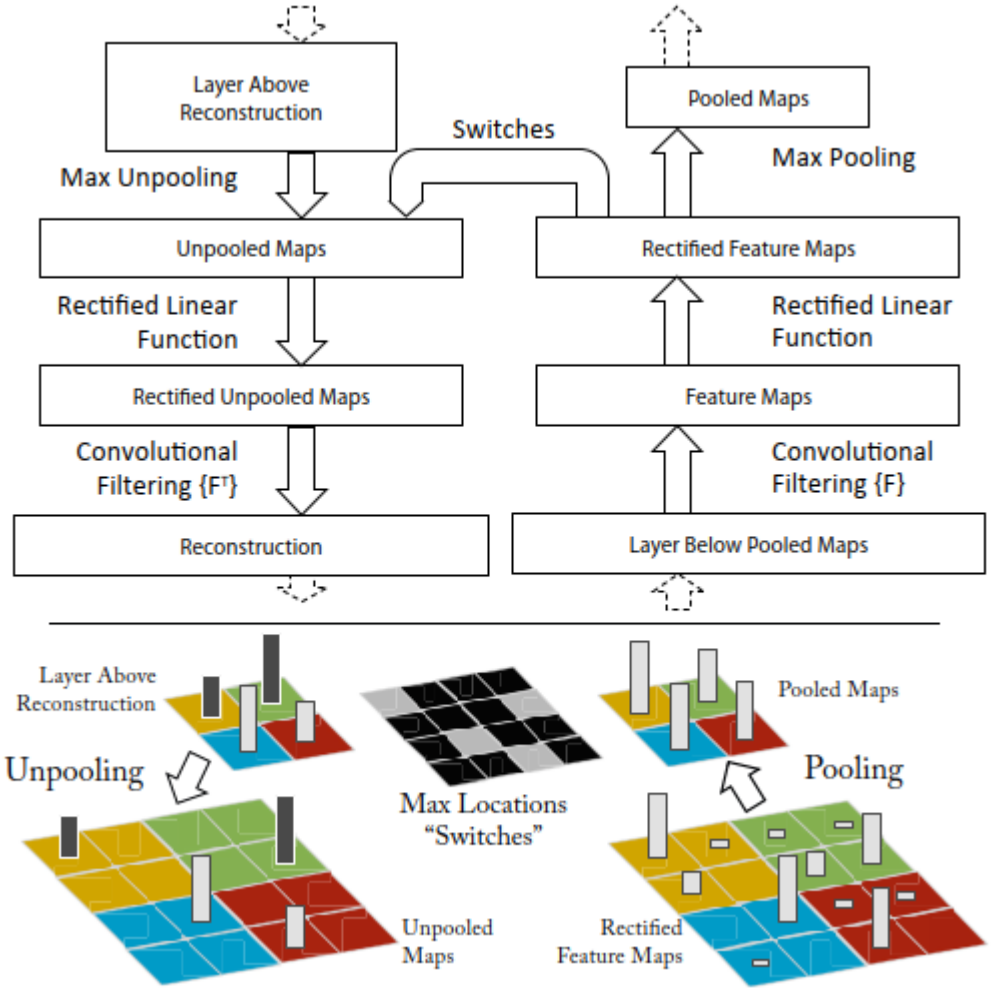
Image source: https://towardsdatascience.com/computer-vision-part-2-8e07029955ee

DenseNet 121 for dog breed recognition

Image source:
https://towardsdatascience.com/computer-vision-part-2-8e07029955ee

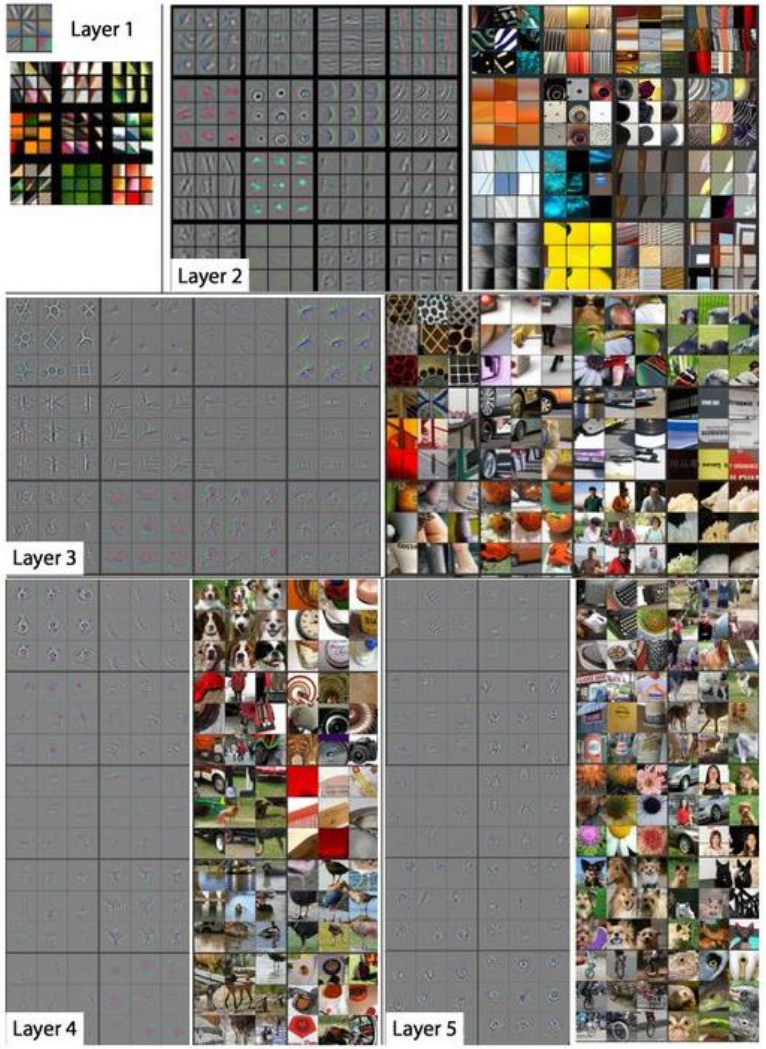


Visualizing which input features lead to strongest activation - Deconvnet



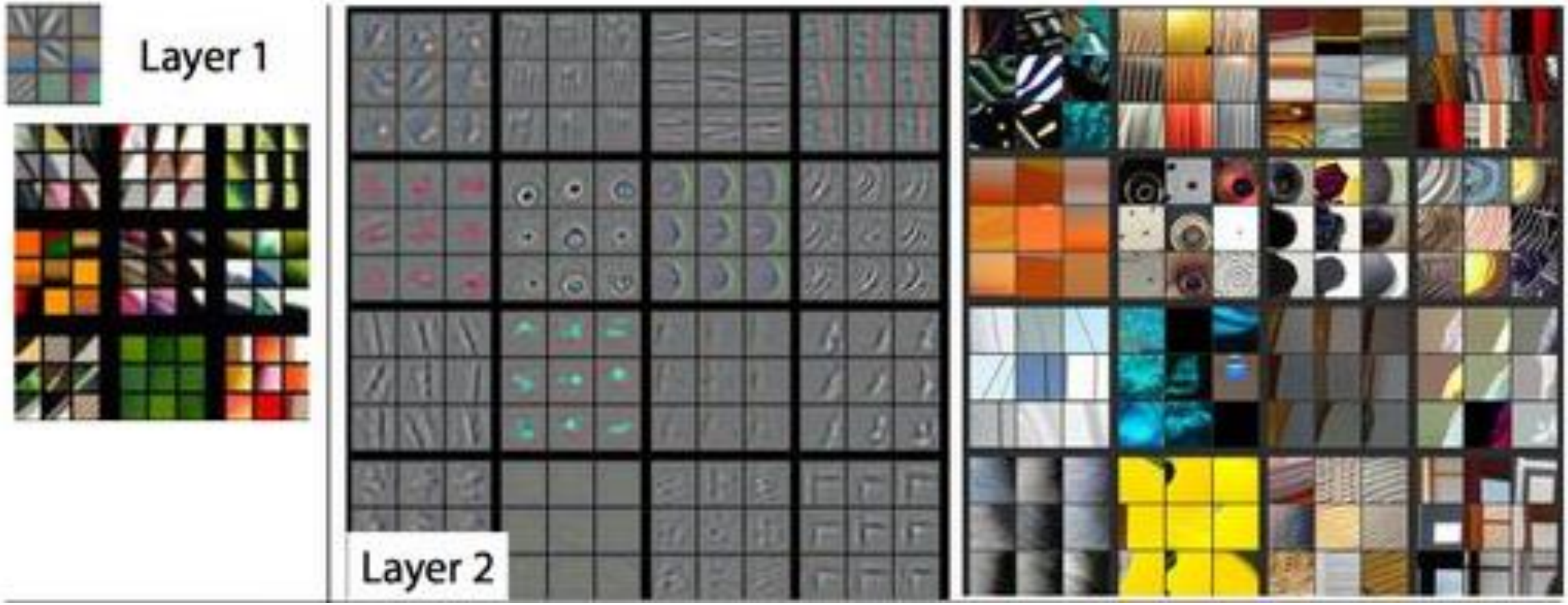
Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath.
Bottom: An illustration of the unpooling operation in the deconvnet, using switches which record the location of the local max in each pooling region (colored zones) during pooling in the convnet. The black/white bars are negative/positive activations within the feature map.
Image source: Zeiler, Matthew D., and Rob Fergus 2014

Visualizing which input features lead to strongest activation

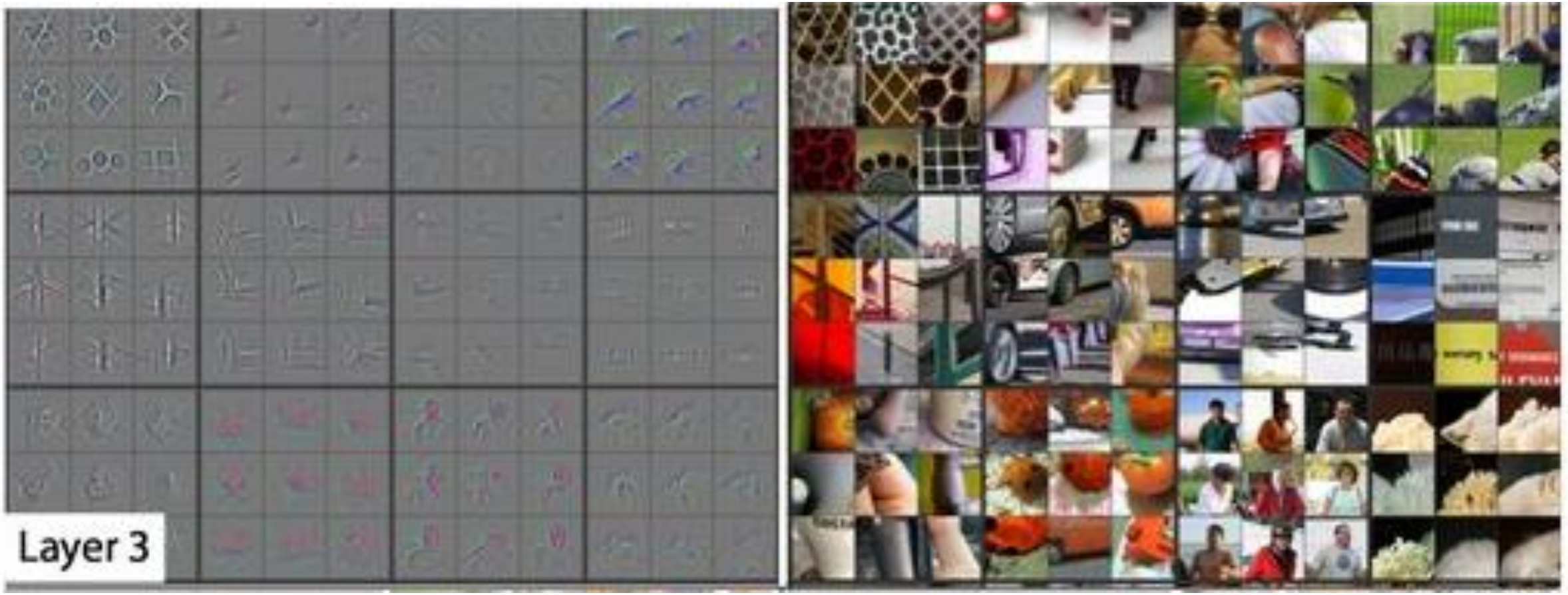


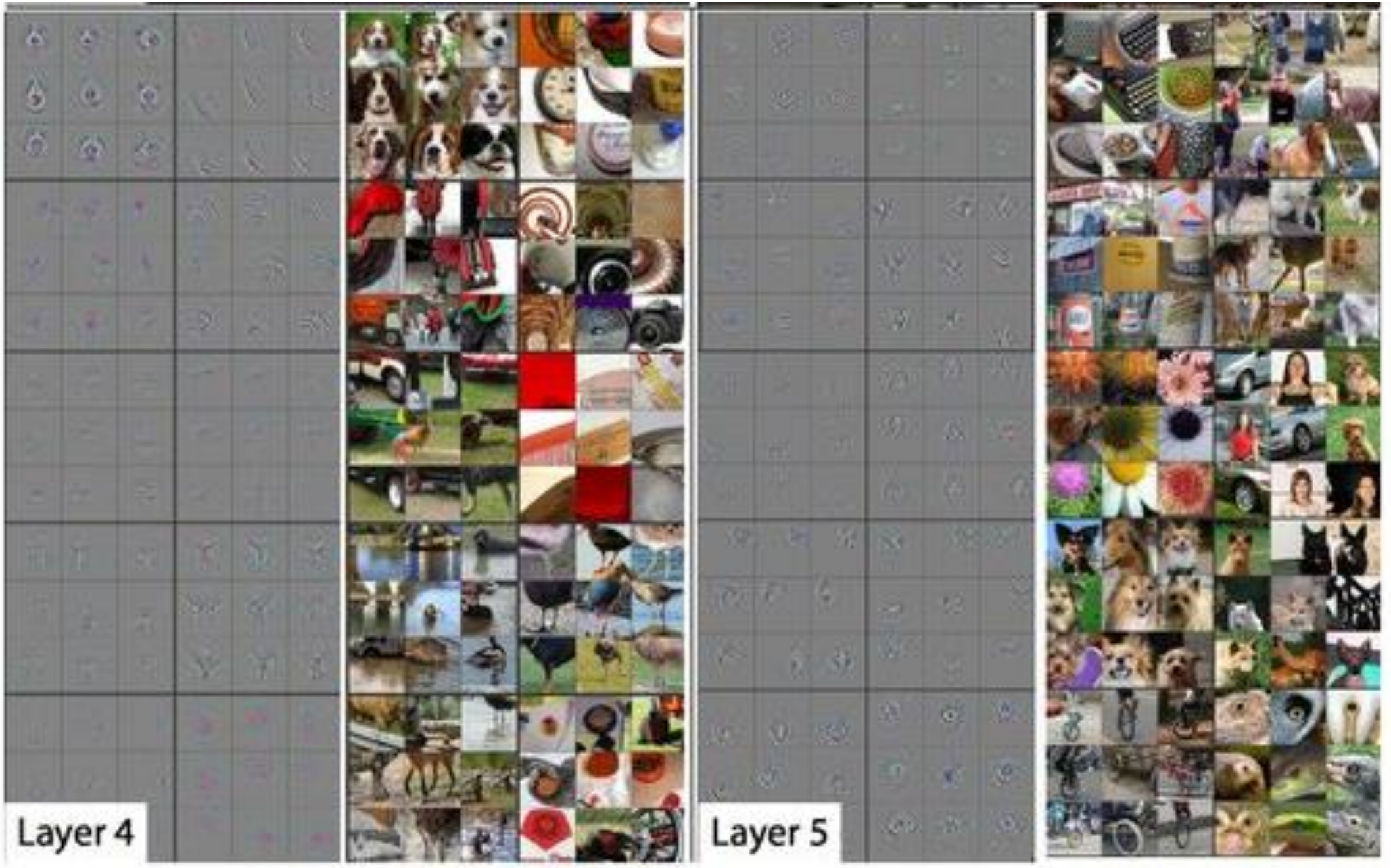
Visualization of features in a fully trained model. For layers 2-5 the top 9 activations in a random subset of feature maps across the validation data are shown, projected down to pixel space using the Deconvnet approach.
Image source: Zeiler, Matthew D., and Rob Fergus 2014

Visualizing which input features lead to strongest activation – layers 1 and 2



Visualizing which input features lead to strongest activation – layer 3




Layer 4
Layer 5

How are CNNs learning?

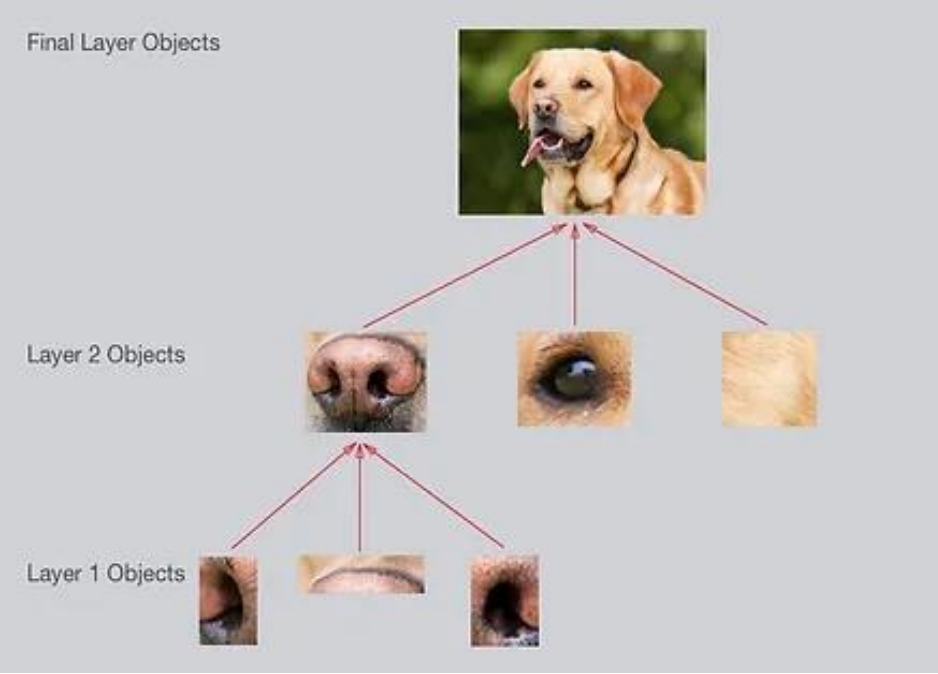
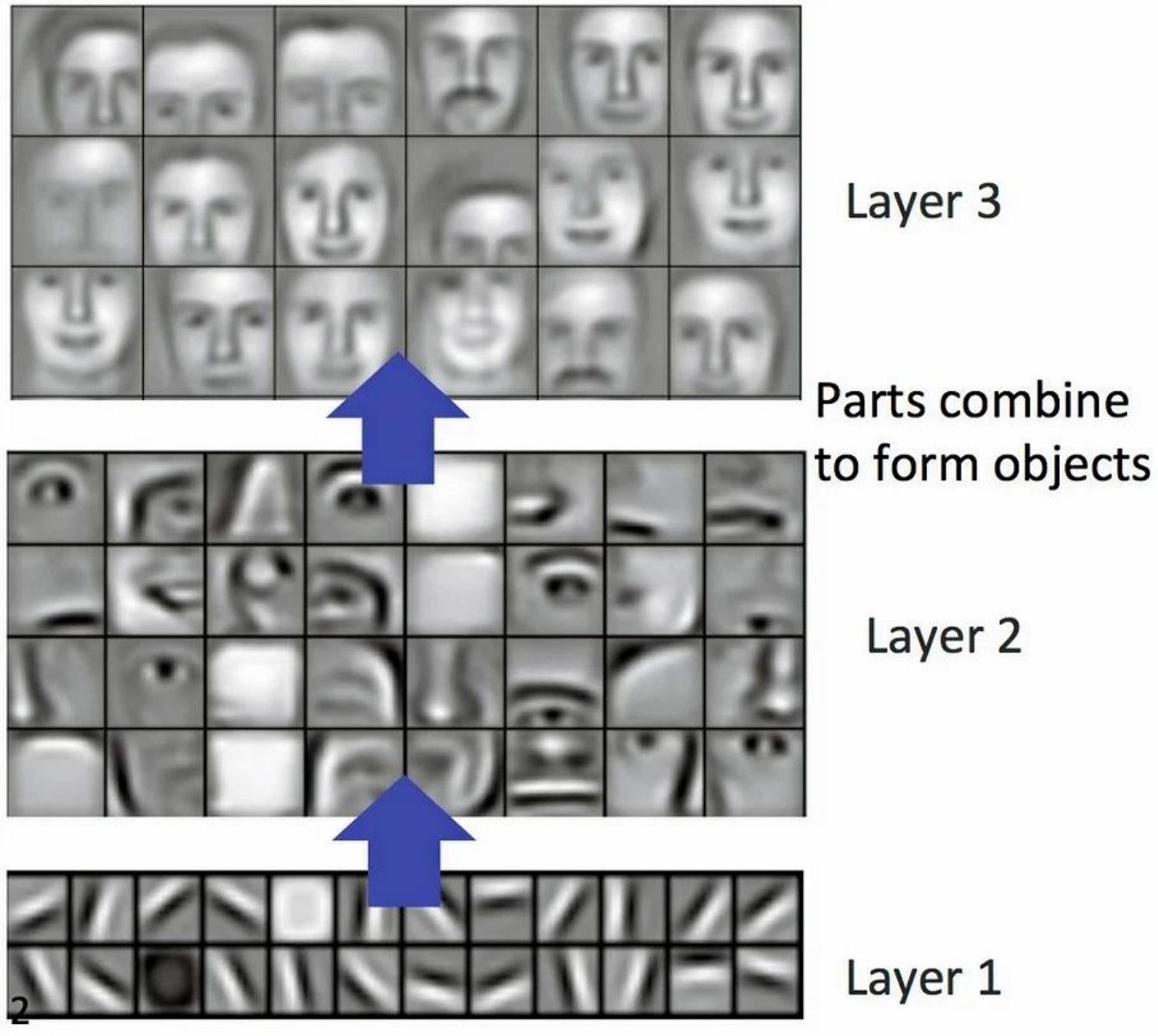


Image source:
https://www.nibivid.com/post/convolutional-neural-net-part-1



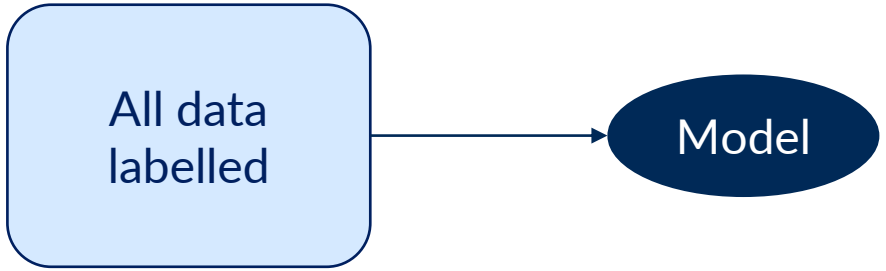
When are CNN features generalize well?

- Representation that generalize well are those that are **robust to variations** in the input data and can capture the essence of the data without fitting too closely to the specific details of the training set.
- CNN properties that improve generalization
  - Hierarchical feature learning
  - Weight sharing
- Further techniques for improving generalization
  - Regularization techniques
  - Data augmentation
  - Diverse training data
  - Transfer learning

Different approaches for representation learning

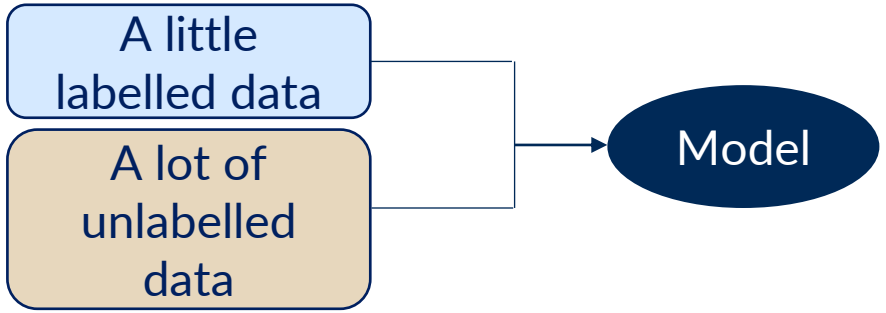
Different types of representation learning

Supervised learning

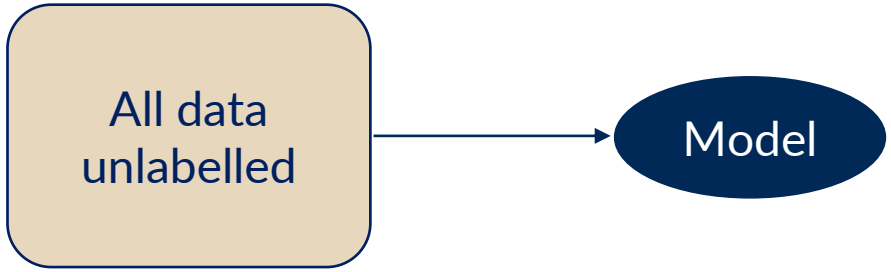


LDA, MLP, CNN

Semi-supervised learning



Unsupervised learning



PCA

Autoencoders for unsupervised feature learning

- The **encoder** transforms input x into the latent vector h: $h = f(x)$
- The **decoder** reconstructs the input from h: $\hat{x} = g(h)$
- The training process involves updating the weights of the encoder and decoder networks according to the reconstruction loss:

$$\mathcal{L}(x, \hat{x}) = \mathcal{L}\Big(x, g\big(f(x)\big)\Big)$$

- Possible reconstruction losses: L1, MSE, binary cross entropy

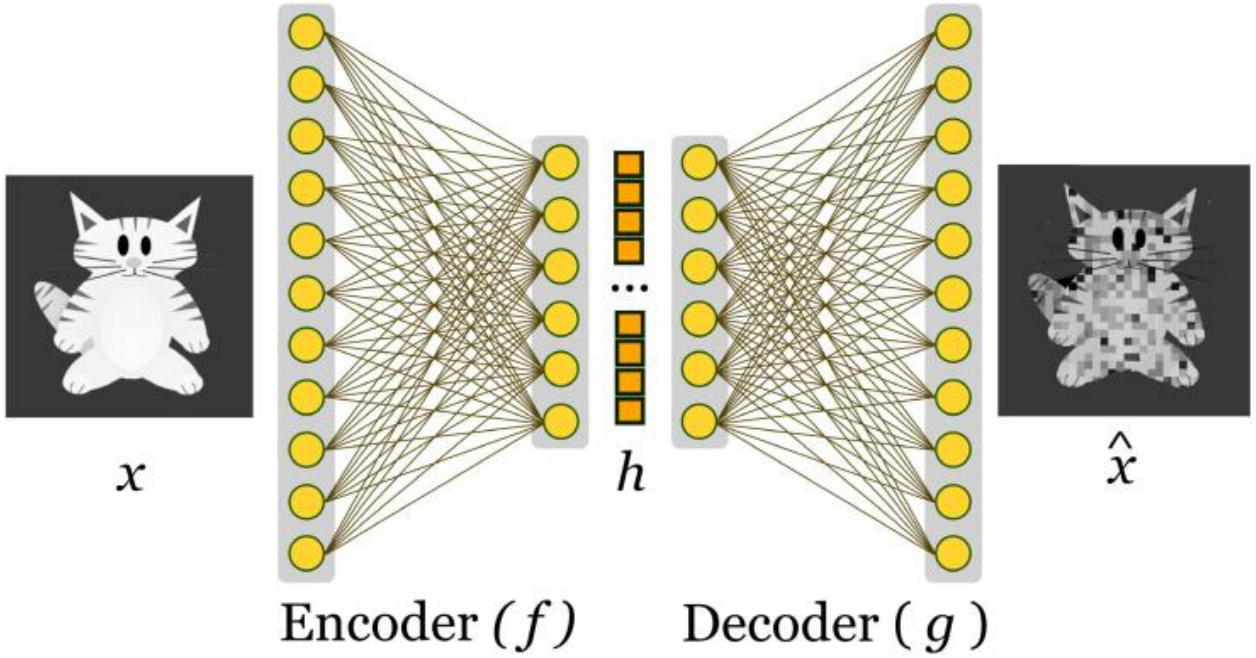


Image source: Baghaei et al. 2023

Denoising autoencoder

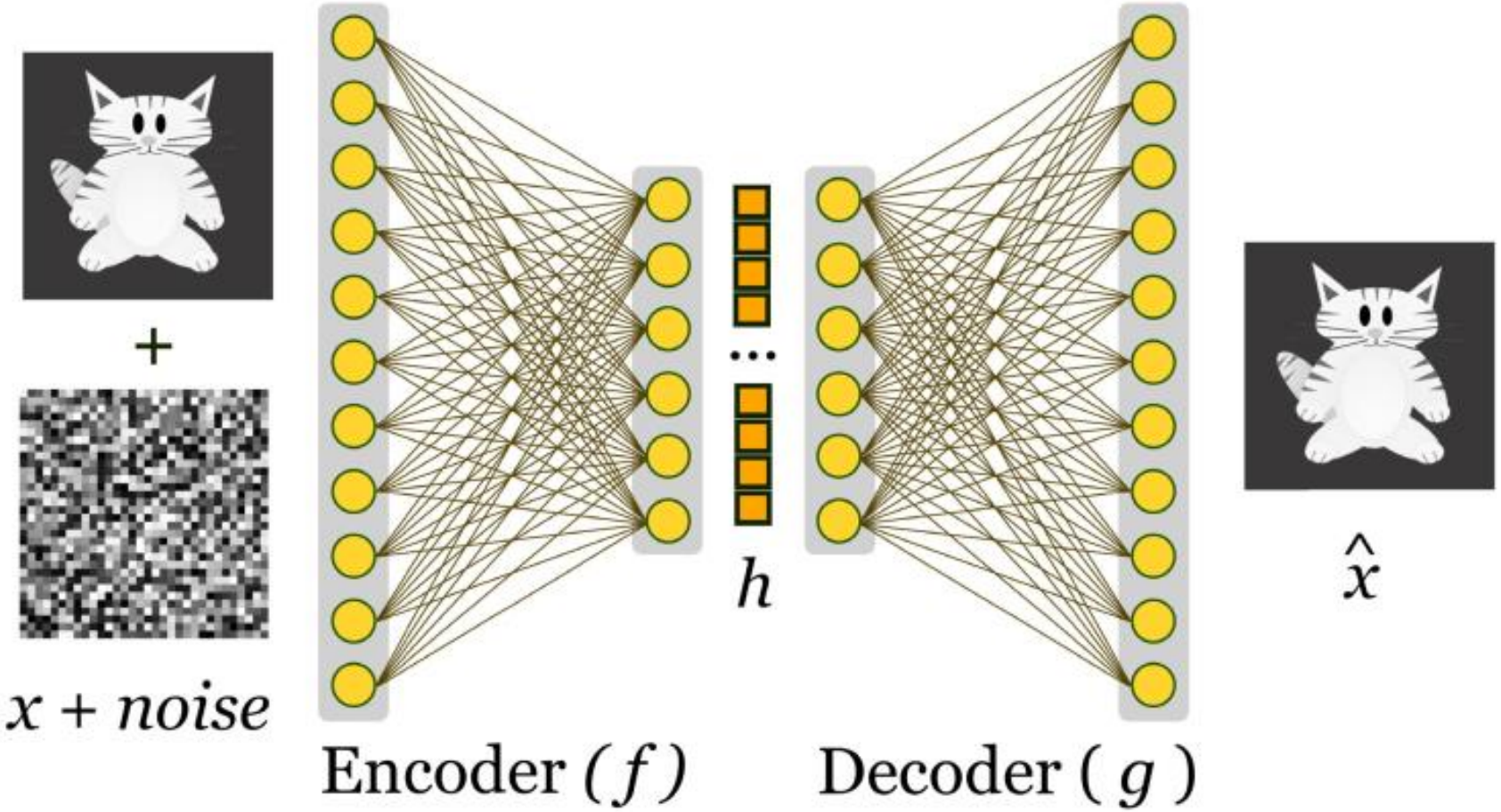
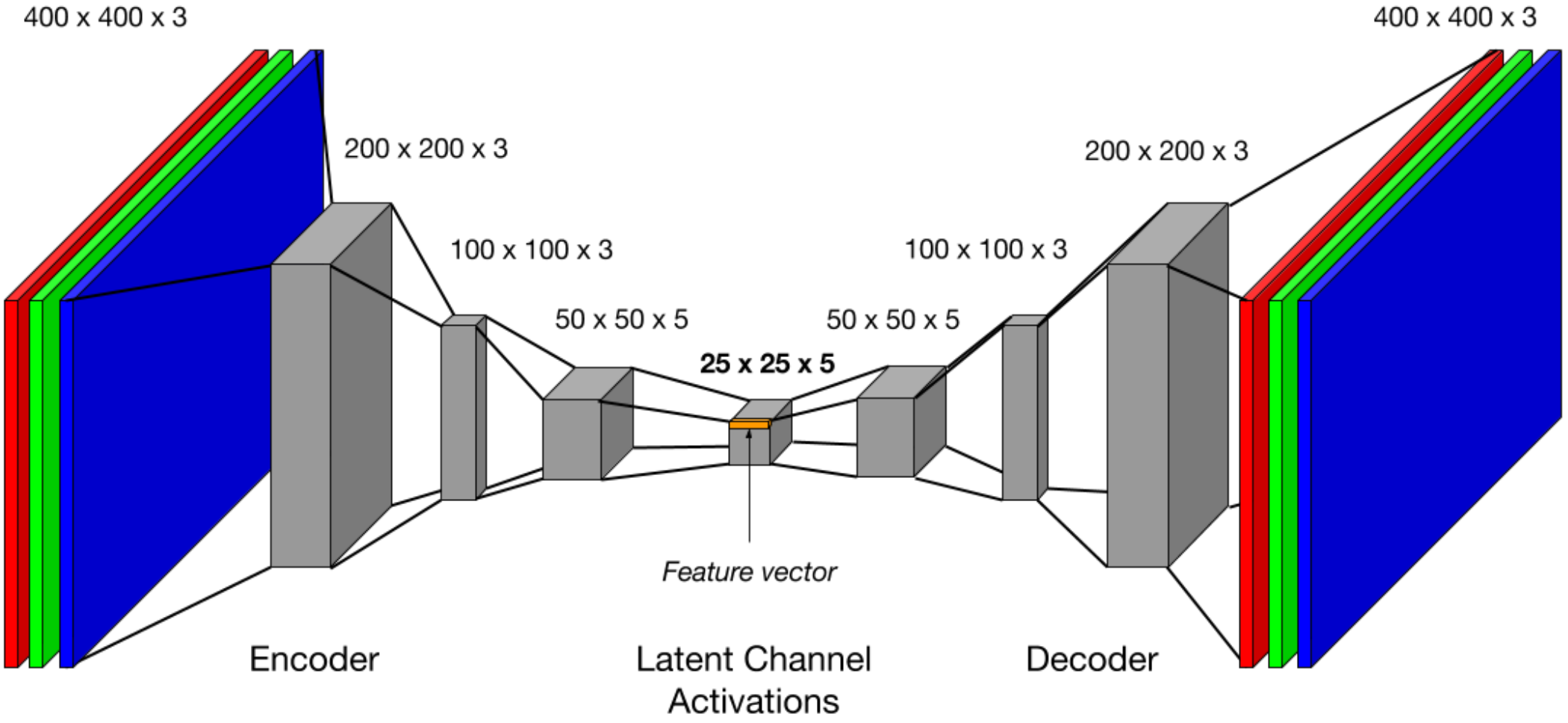


Image source: Baghaei et al. 2023

Convolutional autoencoder



Deep semi-supervised learning approaches

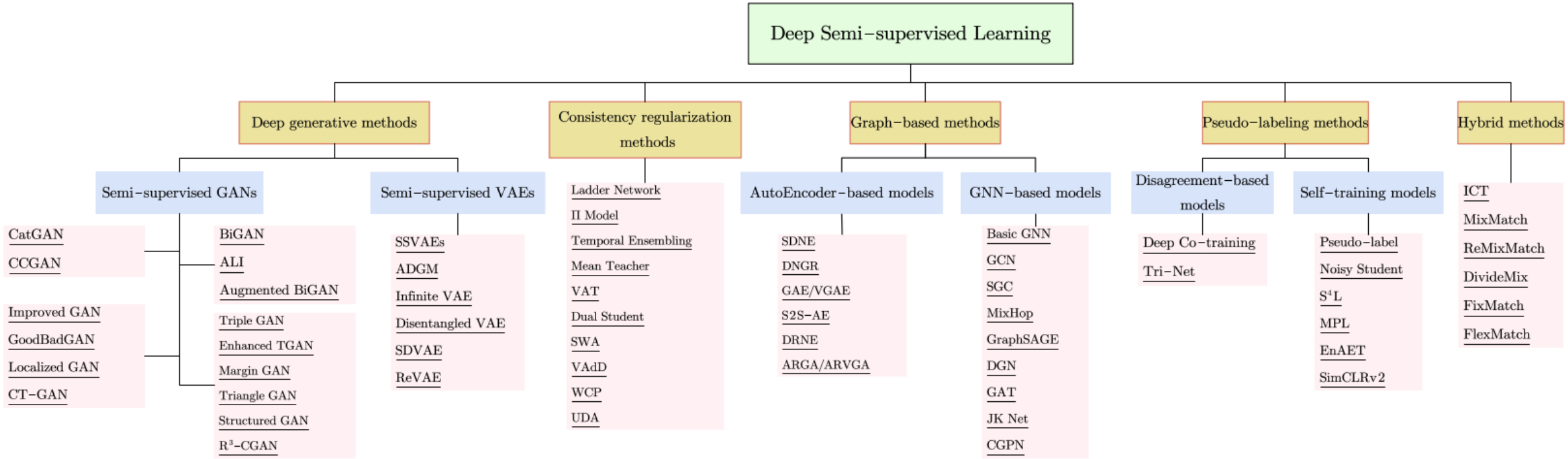
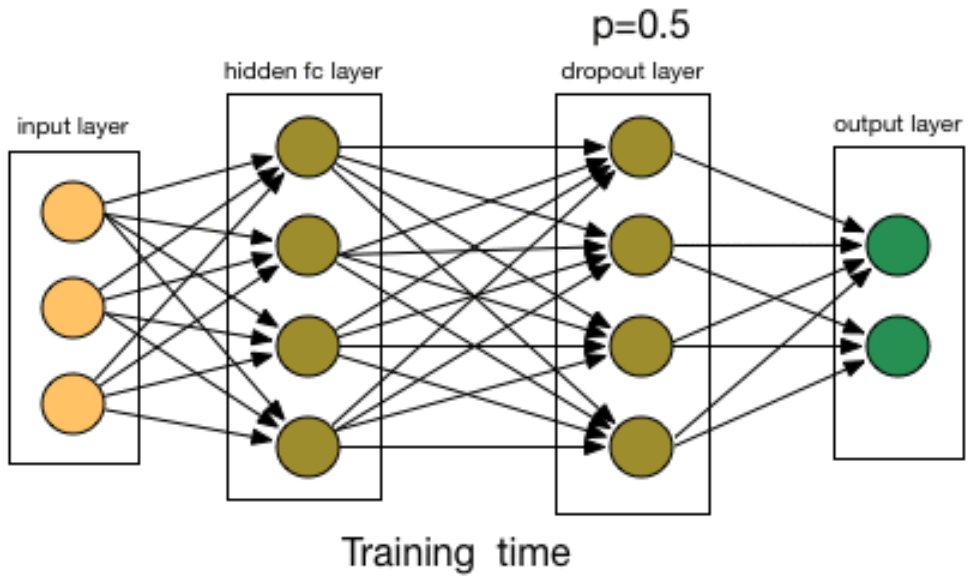


Image source: Yang et al. 2023

Consistency regularization methods

- Consistency regularization is a commonly-used technique for semi-supervised and self-supervised learning.
- The main idea: the model should output the same class distribution for an unlabeled example even after it has been perturbed.
- Approaches for constructing constraints
  - input perturbation
  - weights perturbation
  - layer perturbations



Dropout
Image source: https://primo.ai/index.php?title=Dropout

Ladder network for consistency regularization

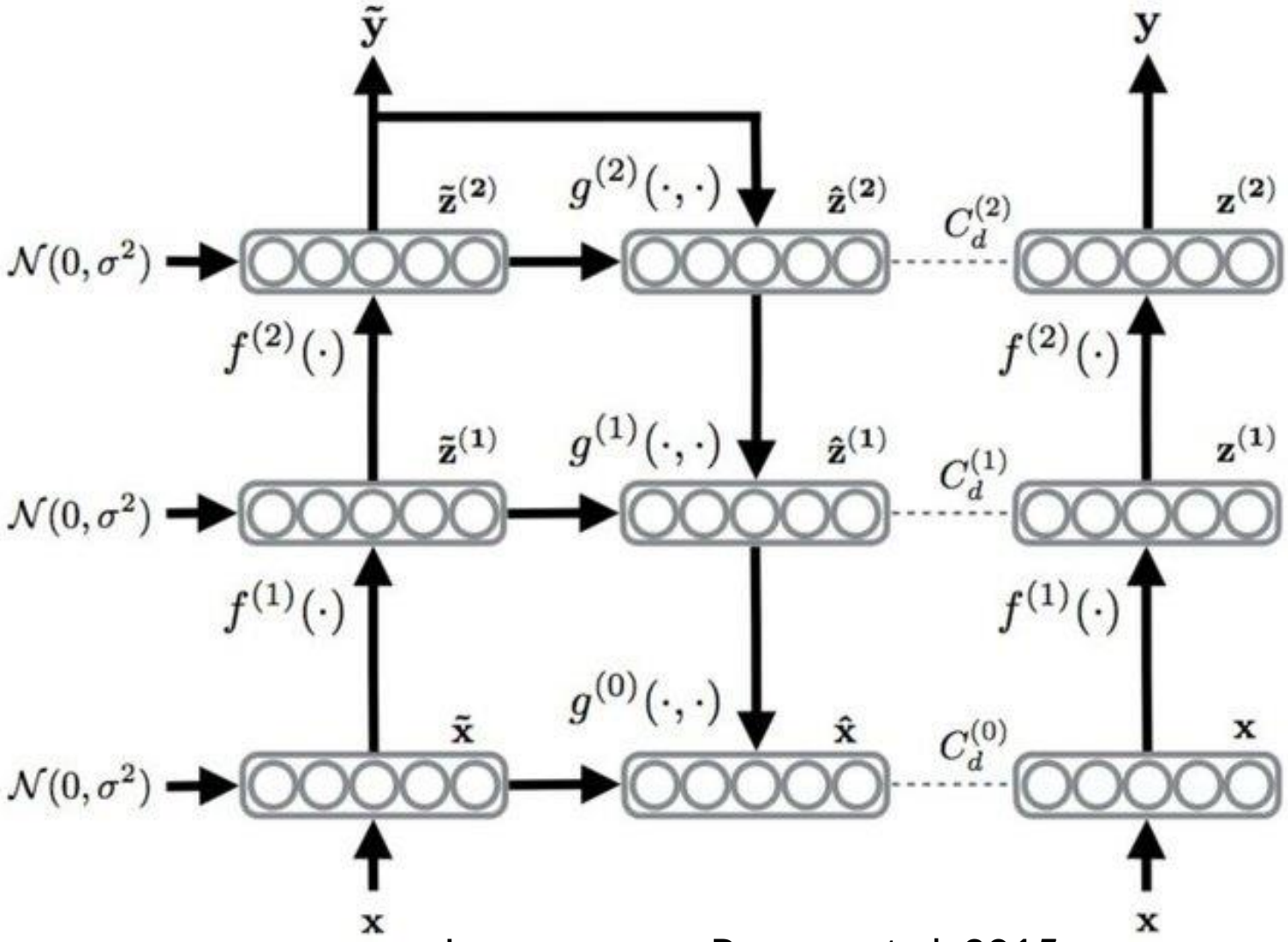


Image source: Rasmus et al. 2015

Consistency regularization methods

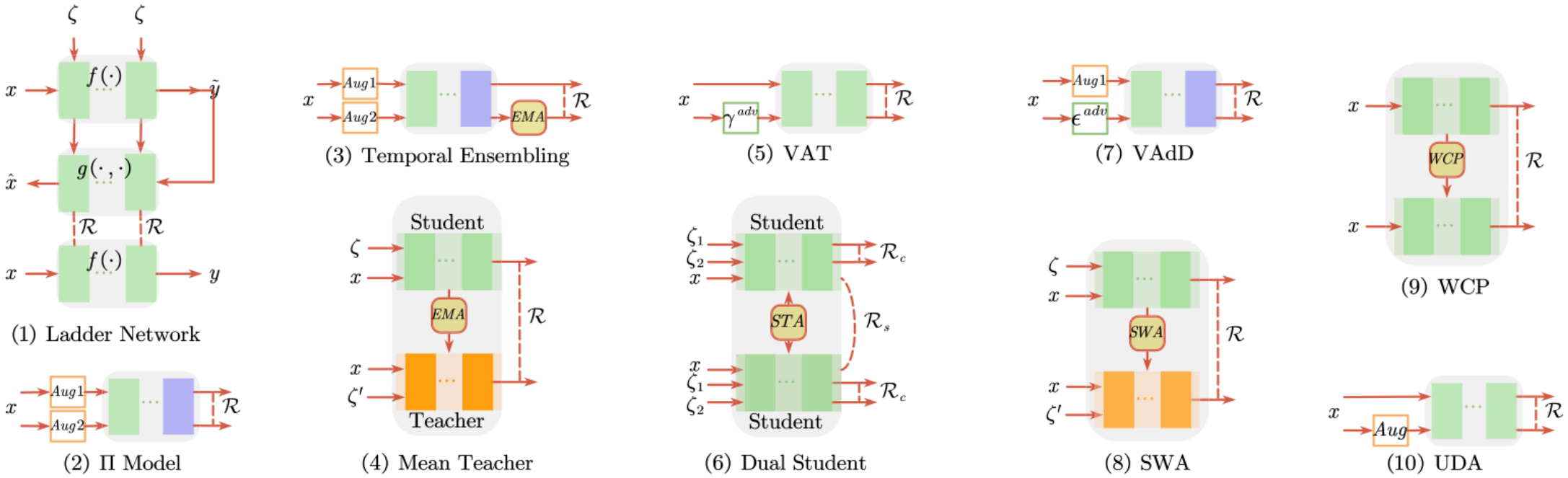
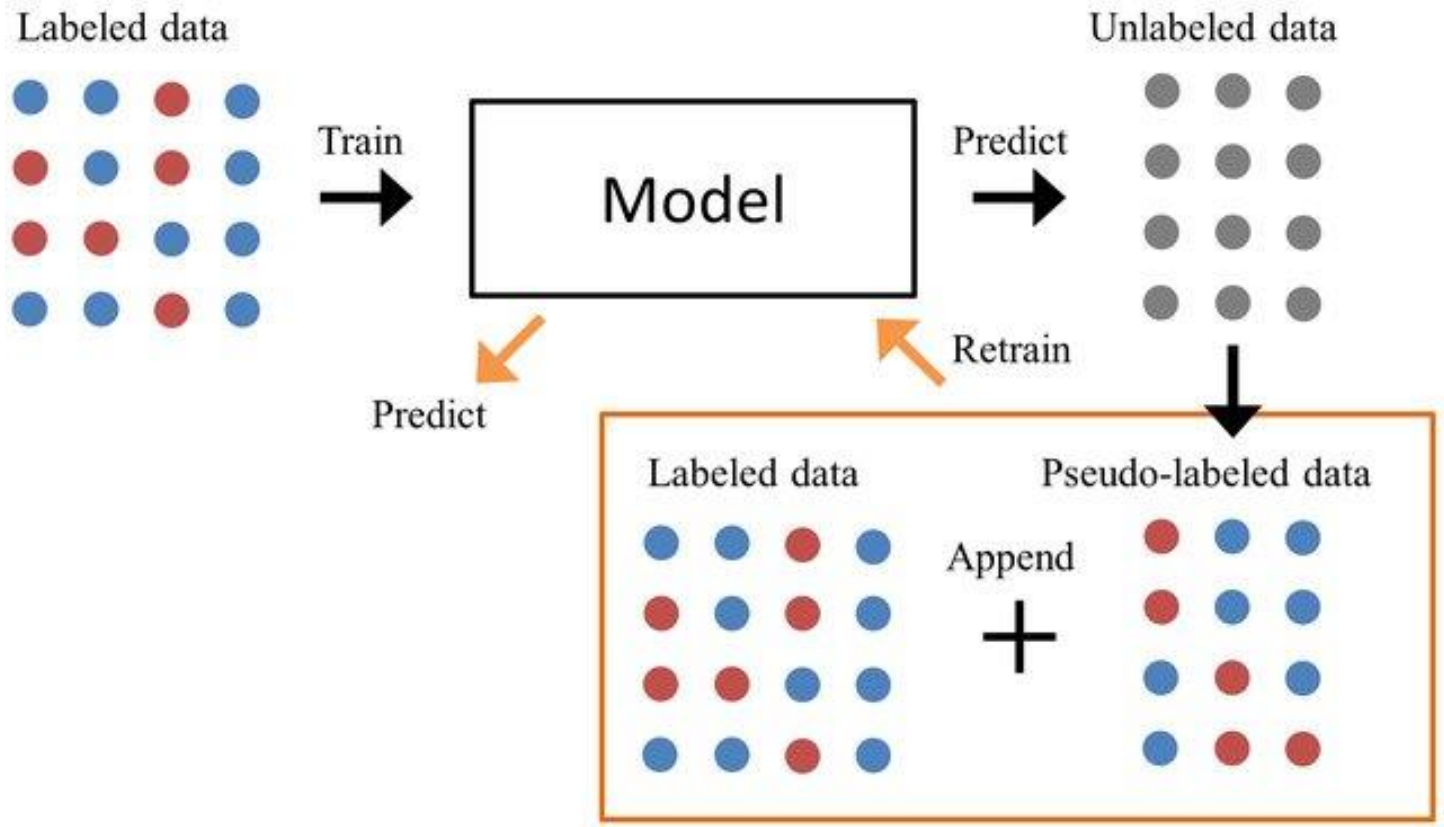


Image source: Yang et al. 2023

Pseudo-labeling

- Pseudo-labeling is the process of adding **confident predicted test data** to your training data to increase the amount of training data



Pseudo-labeling methods

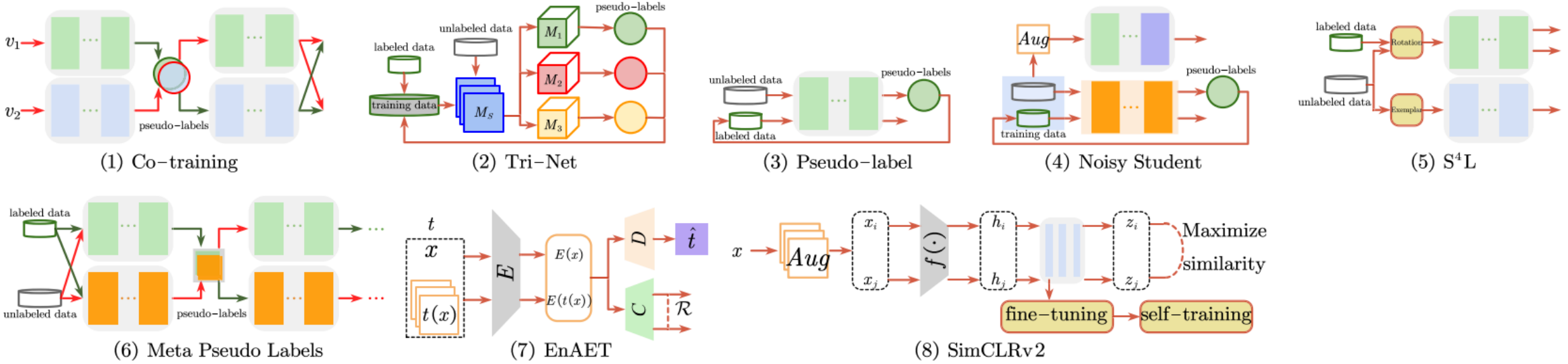


Image source: Yang et al. 2023

Graph-based semi-supervised methods

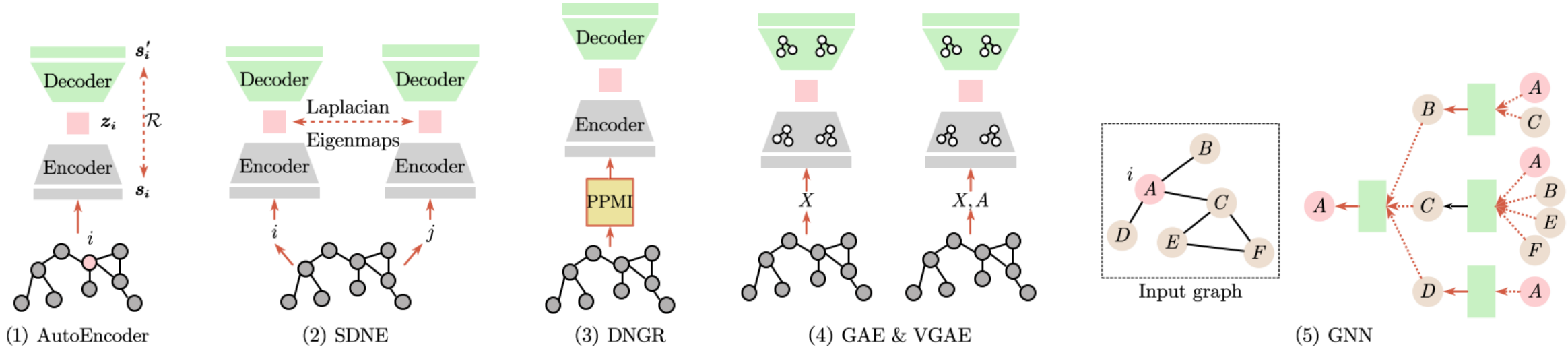
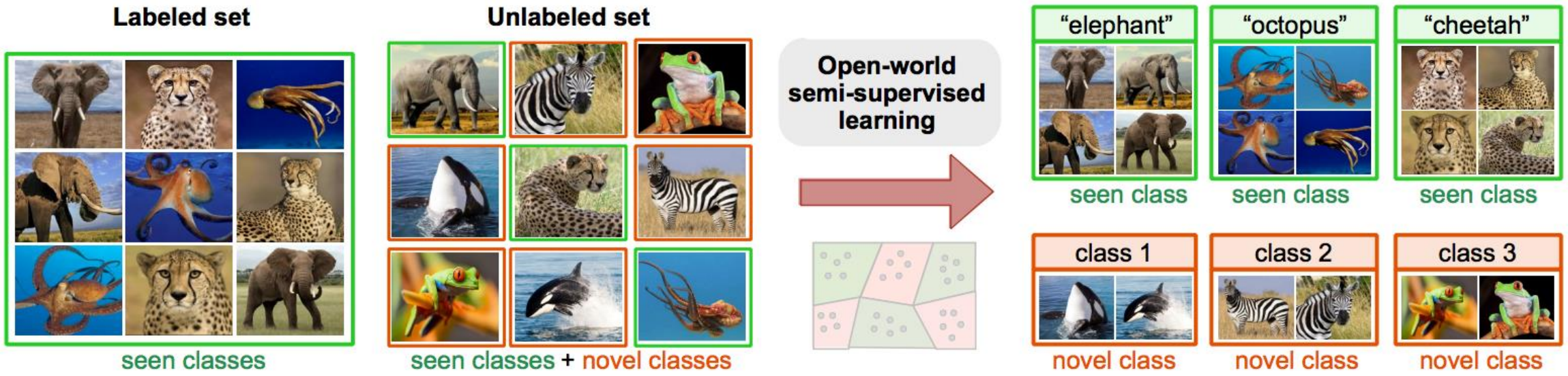


Image source: Yang et al. 2023

- Graph-based semi-supervised learning used the intrinsic structure of data represented as graphs to infer the labels of unlabeled data.

Open-world semi-supervised learning



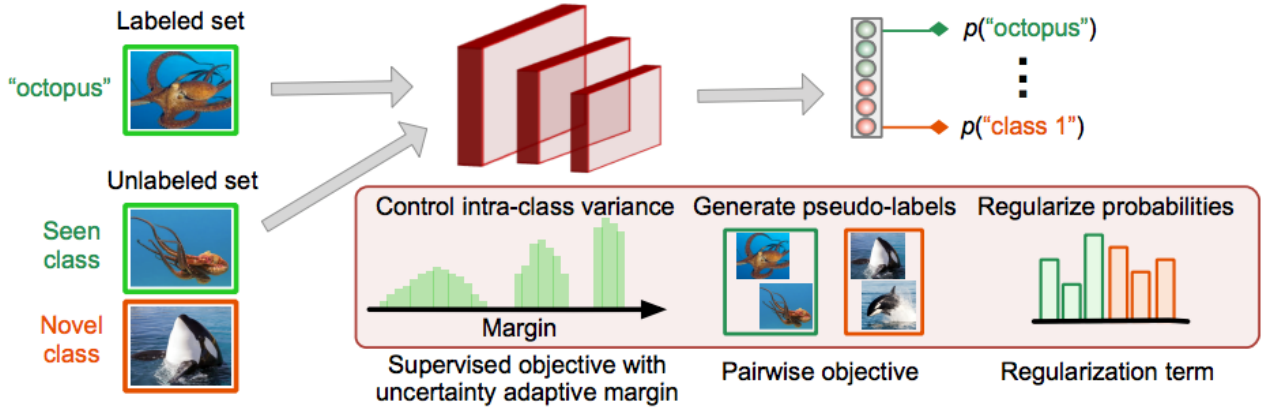


Image source:
http://snap.stanford.edu/orca/owssl_setting.png

Transfer learning

Transfer learning

- The idea of transfer learning is to take a pre-trained model, which has been trained on a large and general dataset, and repurpose it for a different but related problem.
- It involves **leveraging the learned representation from one task** and applying it to another, often with less data or computational resources.

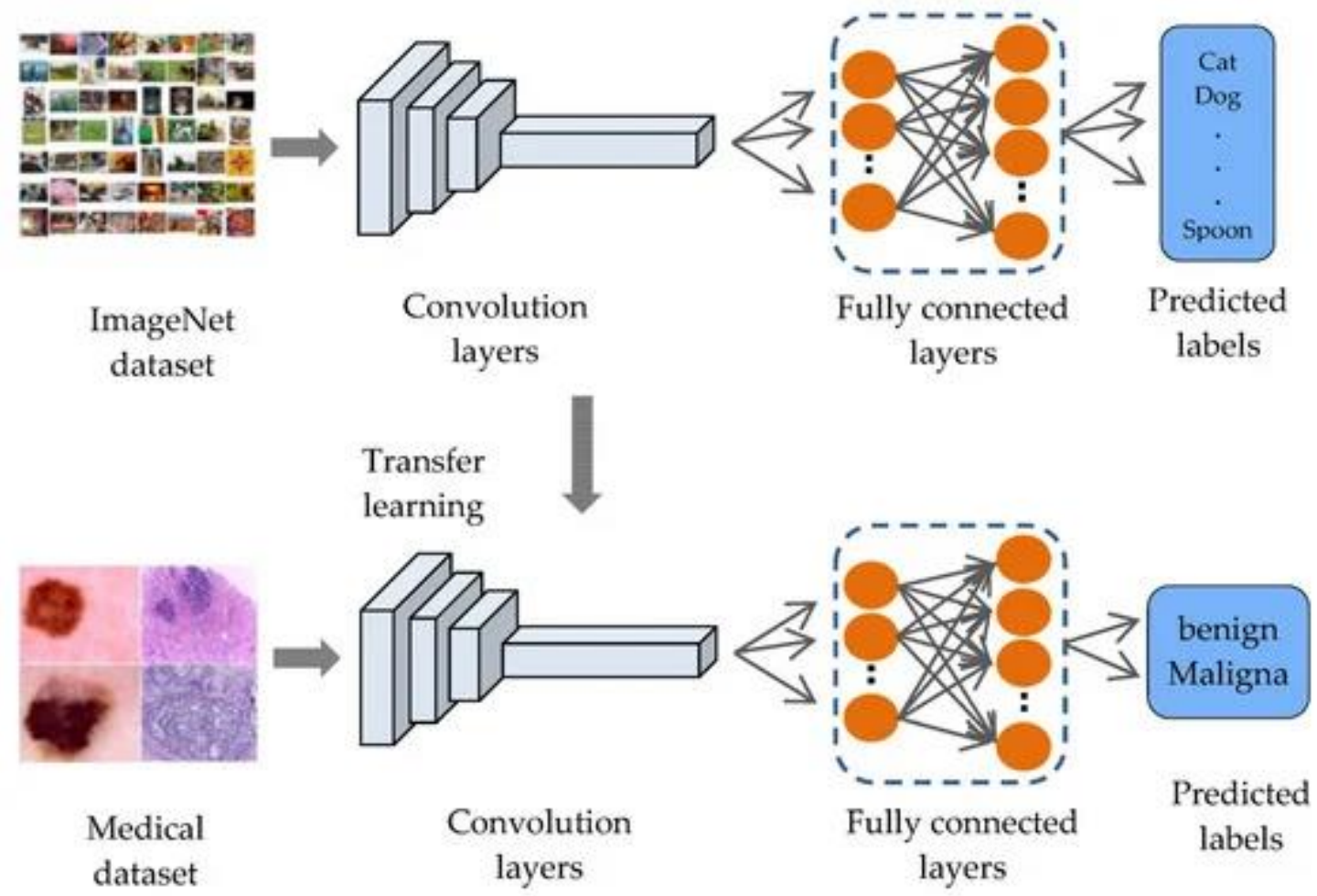


Image source: Mukhlif et al., 2023

Benefits of transfer learning

- **Knowledge Transfer:** Transfer learning allows the transfer of knowledge from one domain to another. This is particularly useful when the new domain has limited labeled data.
- **Efficiency:** It reduces the need for extensive training as the model has already learned patterns from a large dataset. This saves time and computational resources.
- **Feature Reuse:** The lower layers of a neural network capture universal features like edges and textures, which are useful across various tasks. Transfer learning exploits these features without the need for re-learning.
- **Adaptability:** It provides flexibility to adapt a model to new tasks with some fine-tuning, making it possible to achieve high performance even on tasks that are significantly different from the original task the model was trained on.
- **Performance Boost:** For many tasks, models trained with transfer learning outperform those trained from scratch, especially when the available data is scarce.

Transfer learning

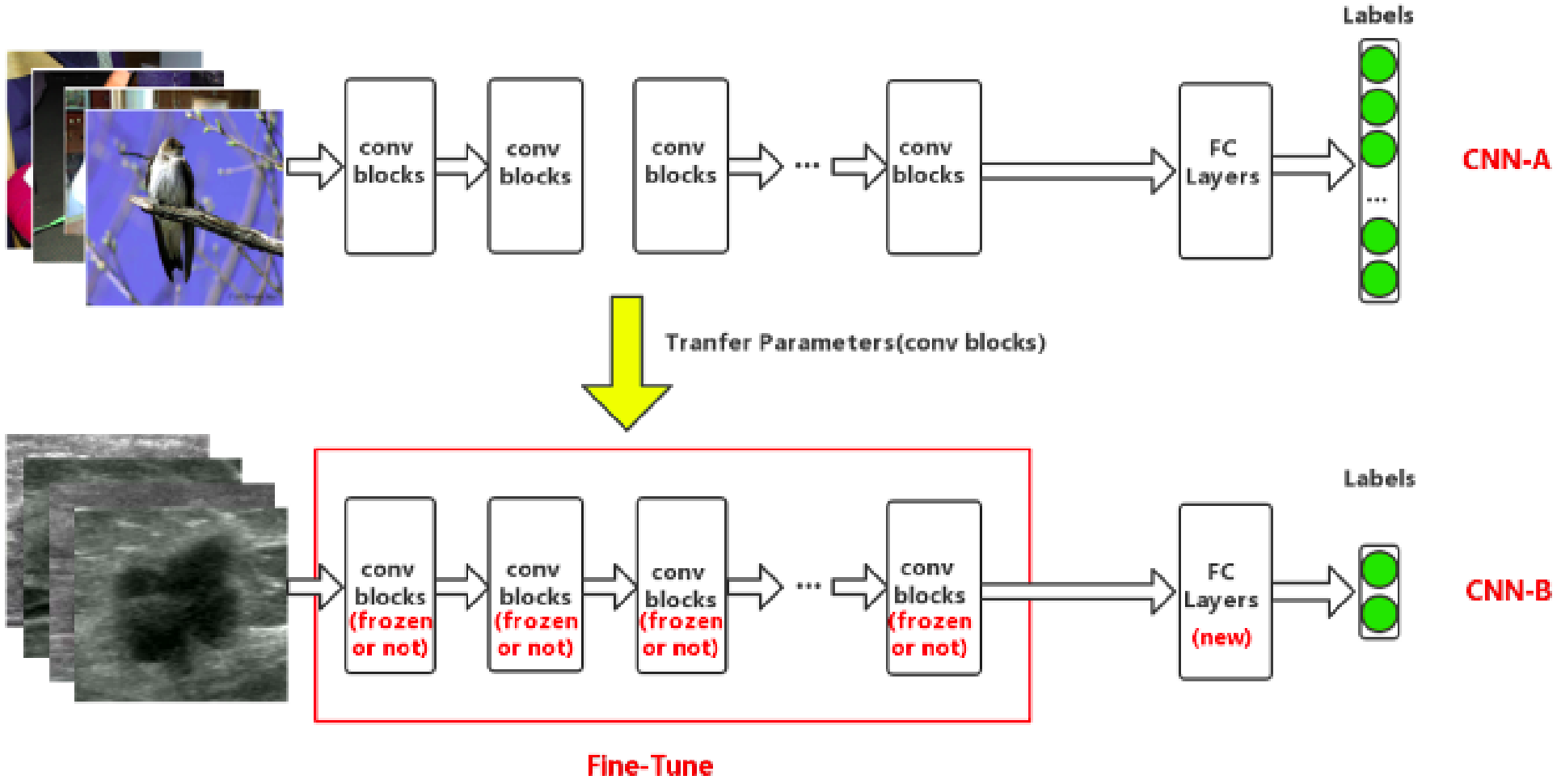


Image source: https://neptune.ai/blog/transfer-learning-guide-examples-for-images-and-text-in-keras

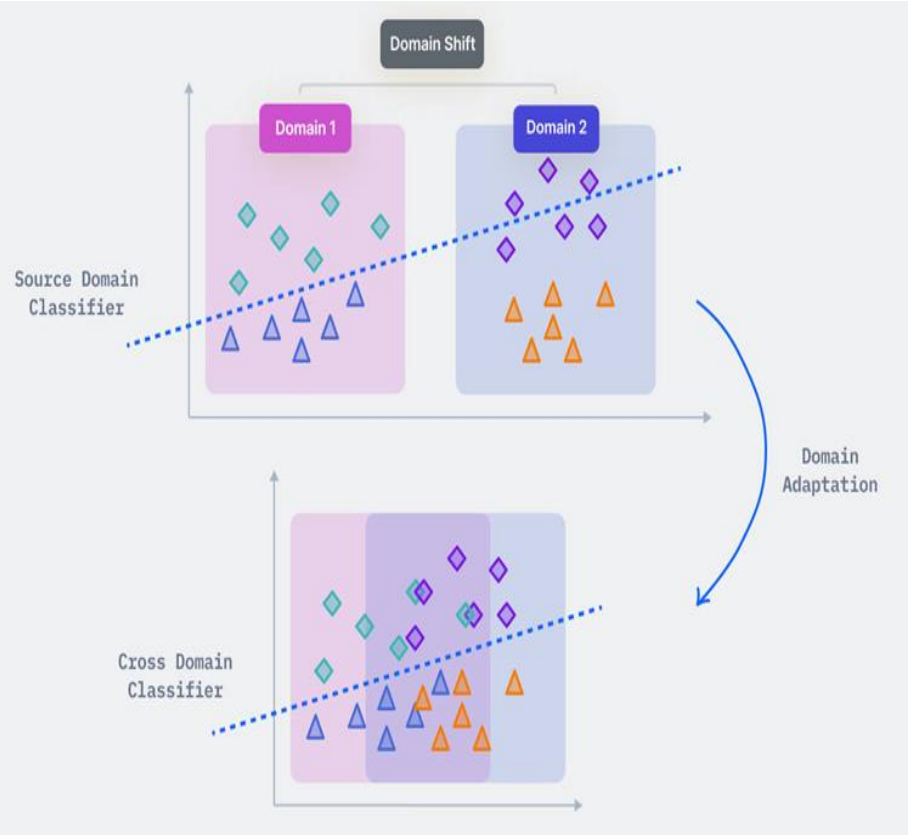
How to apply transfer learning?

- **Data Augmentation:** Apply data augmentation techniques to increase the diversity of your dataset without actually collecting new data.
- **Select Base Model:** Choose a pre-trained model that has been trained on a large dataset and is capable of extracting robust features. If available, use a model trained with a similar dataset.
- **Modify for New Task:** Depending on your task, you may need to modify or replace the top layers of the model to suit your specific needs.
- **Define training parameters**: Define the learning rate and loss function.
- **Freeze Layers:** Freeze the layers of the base model that you do not want to train. Typically, these are the early layers that capture universal features.
- **Train Model:** Train the model on your dataset. Only the unfrozen layers will be updated during training.
- **Fine-Tune:** Optionally, after the initial training, you can unfreeze more layers and continue training for fine-tuning.

Domain adaptation

Domain adaptation

- Domain adaptation is a subset of transfer learning that specifically addresses the scenario where the **source** and **target** tasks are the same, but the **data distribution is different (domain shift)**.
- It is used when you want to adapt a model to perform well on a target domain that it has not seen during training. This is particularly useful when the model needs to **generalize to new environments** or **conditions**.



Domain shifts

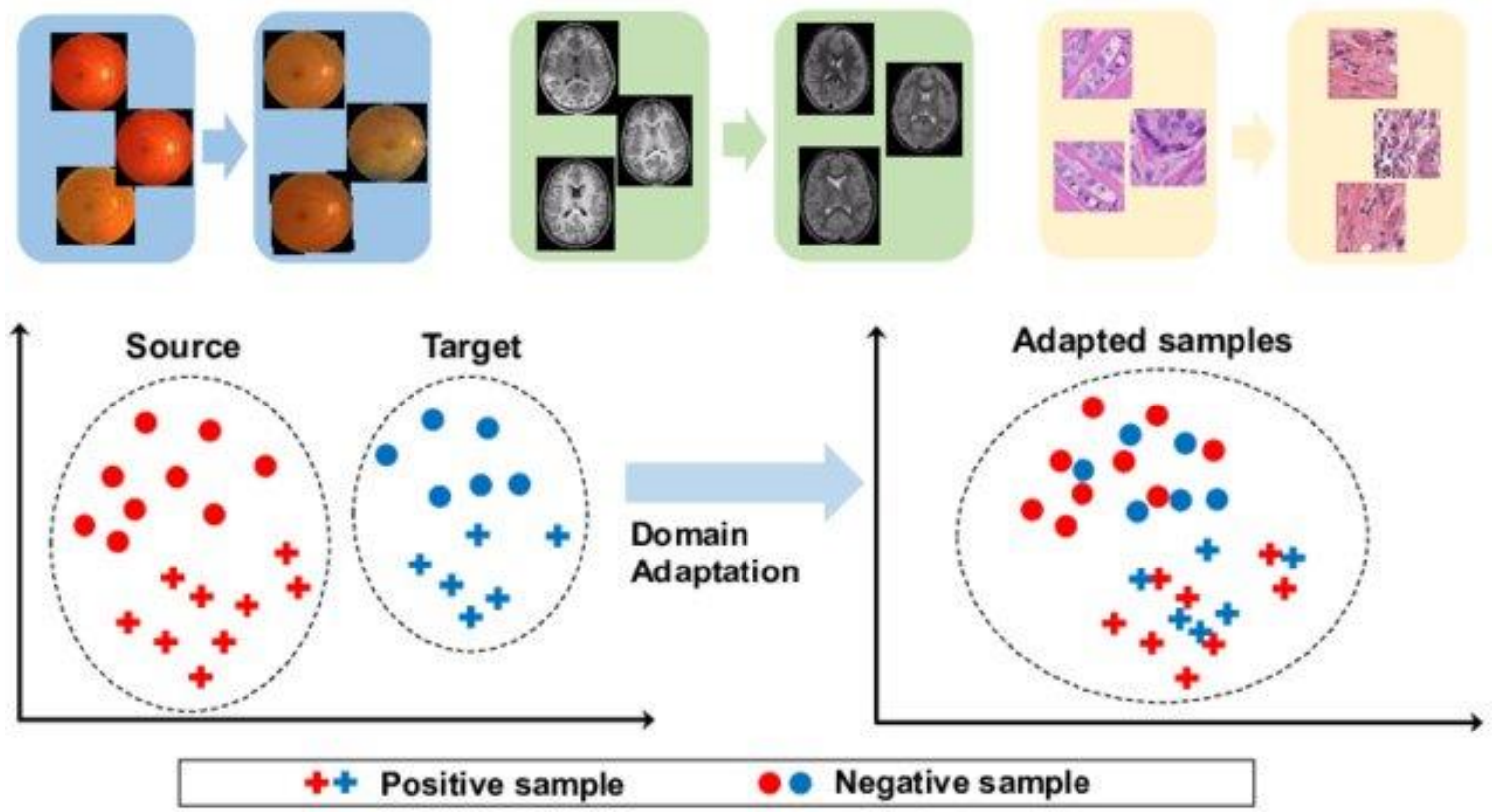


Image source: Guan and Liu 2023

Image source: Pitas 2018

Domain adaptation types

- Unsupervised domain adaptation
  - No labels on target domain
- Semisupervised domain adaptation
  - Few labeled data on target domain
- Supervised domain adaptation
  - All target data labelled

Domain adaptation idea

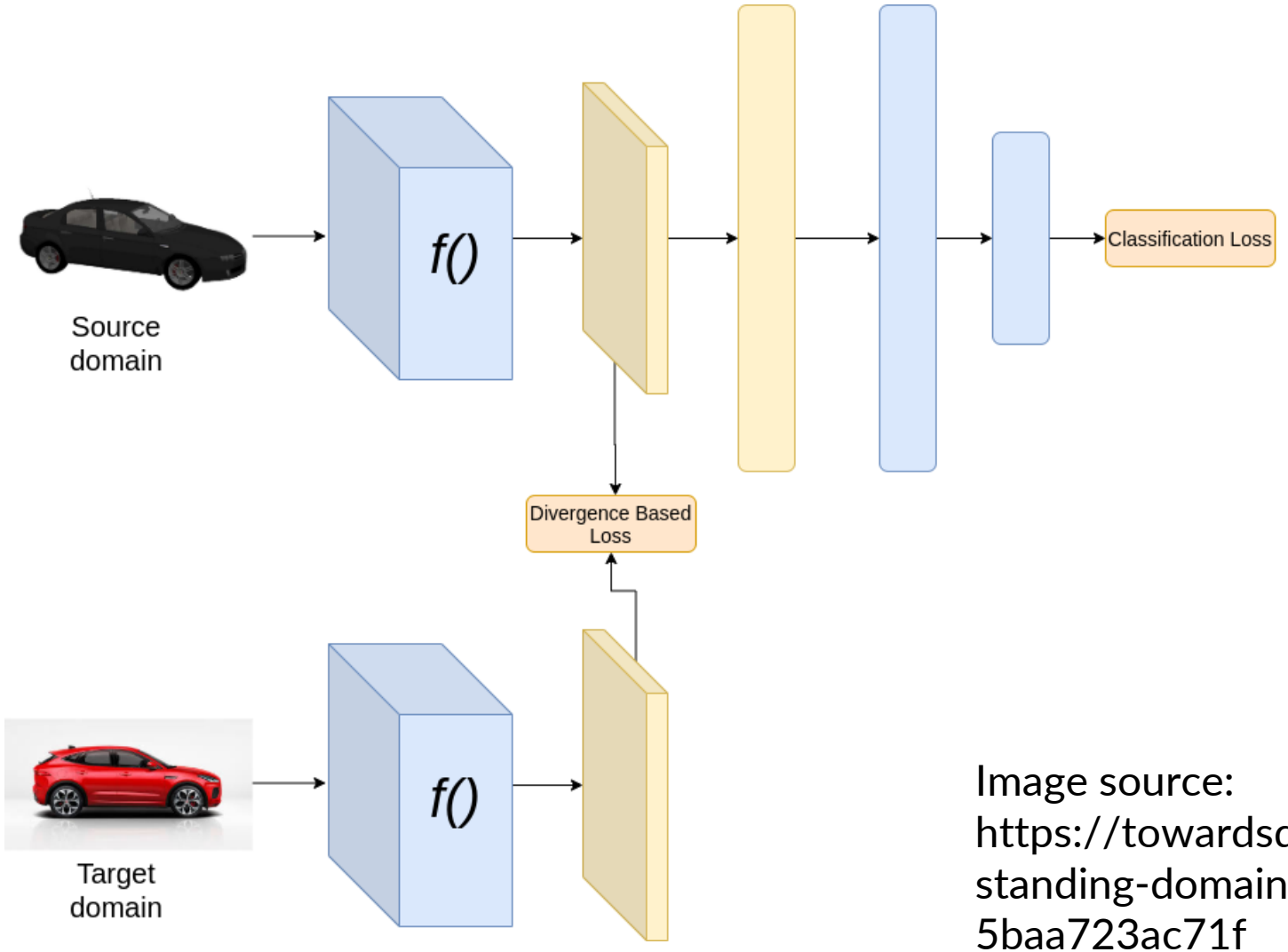


Image source:
https://towardsdatascience.com/understanding-domain-adaptation-5baa723ac71f

Domain adversarial learning

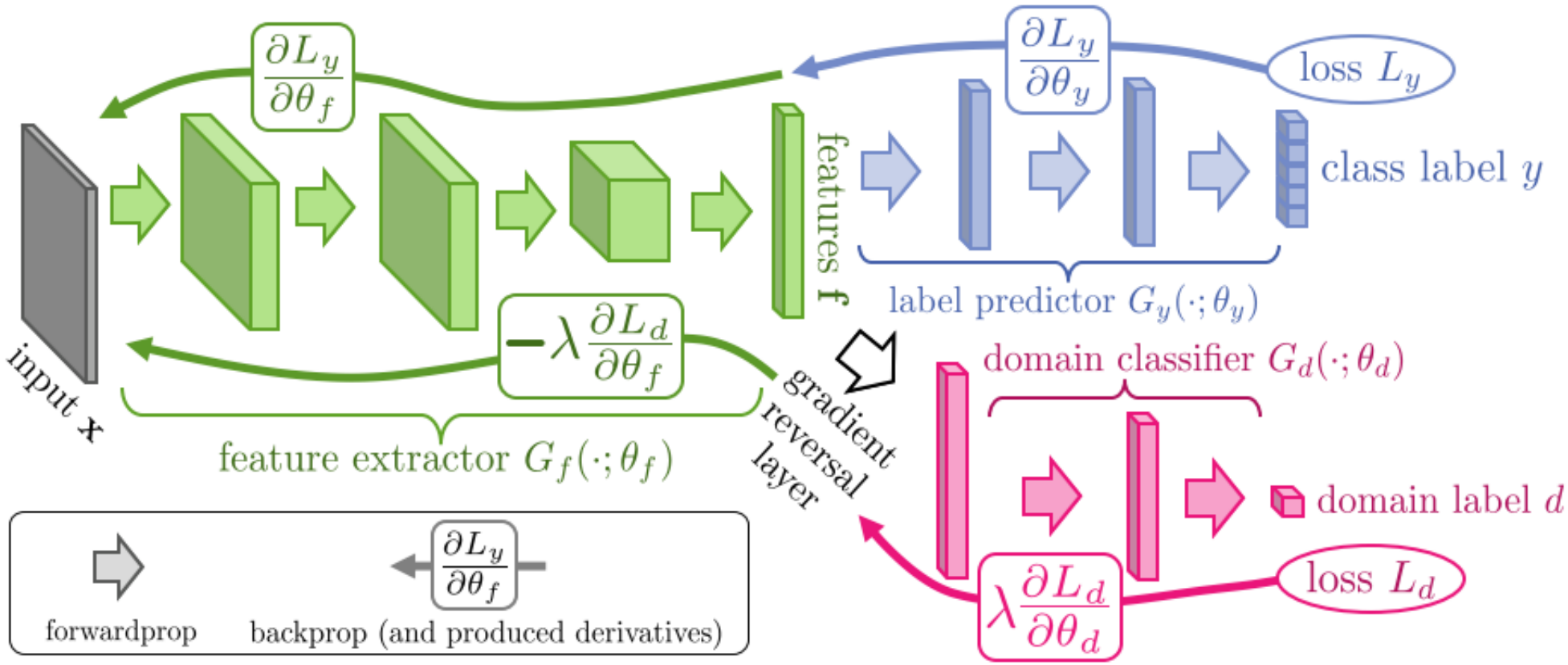


Image source: Nadeem et al. 2016

How to measure divergence between domain?

- Domain adaptation methods typically try to learn representations that **minimize the domain discrepancy**
- For the optimization process, a measure of discrepancy is needed
- **Maximum Mean Discrepancy (MMD)**
  - Given P, Q maximum mean discrepancy is the distance between feature means of P and Q:

$$MMD^2(P,Q) = \|\mu_P - \mu_Q\|^2_{\mathcal{F}}$$

$$MMD^2(P,Q) = \langle \mu_P - \mu_Q,\ \mu_P - \mu_Q \rangle = \langle \mu_P, \mu_P \rangle - 2\langle \mu_P, \mu_Q \rangle + \langle \mu_Q, \mu_Q \rangle$$

  - Practical evaluation of MMD per batch:

$$MMD^2(X,Y) = \underbrace{\frac{1}{m(m-1)} \sum_i \sum_{j \neq i} k(\mathbf{x_i}, \mathbf{x_j})}_{\text{A}} - \underbrace{2\frac{1}{m.m} \sum_i \sum_j k(\mathbf{x_i}, \mathbf{y_j})}_{\text{B}} + \underbrace{\frac{1}{m(m-1)} \sum_i \sum_{j \neq i} k(\mathbf{y_i}, \mathbf{y_j})}_{\text{C}}$$

How to measure divergence between domain?

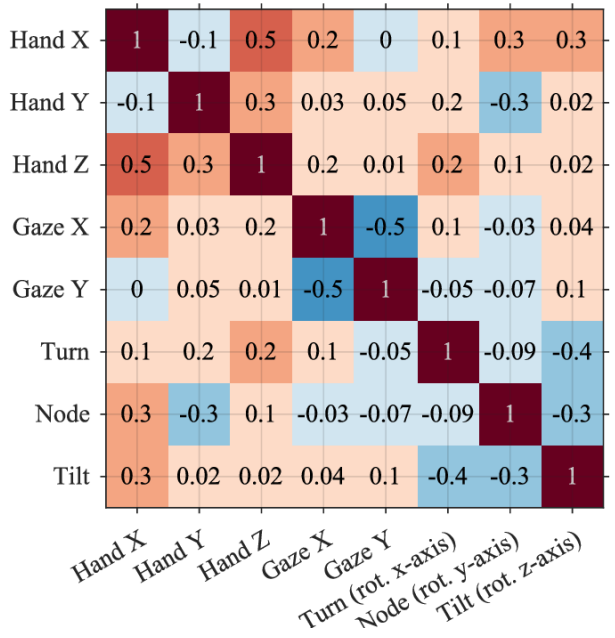
- **Correlation alignment** (coral loss)

$$\min_{F_s, F_t} L_{DM}(F_s, F_t) = \frac{1}{4d^2} \|C_s - C_t\|_F^2,$$

where C_s and C_t denote the feature covariance matrices of the source and target data, d indicates the dimension of the activation features and $||.||_F^2$ denotes the squared matrix Frobenius norm. The C_s and C_t are given by the following equations

$$C_s = \frac{1}{N_s - 1}\left(F_s^T F_s - \frac{1}{N_s}(1^T F_s)^T (1^T F_s)\right),$$

$$C_t = \frac{1}{N_t - 1}\left(F_t^T F_t - \frac{1}{N_t}(1^T F_t)^T (1^T F_t)\right).$$



Deep unsupervised domain adaptation

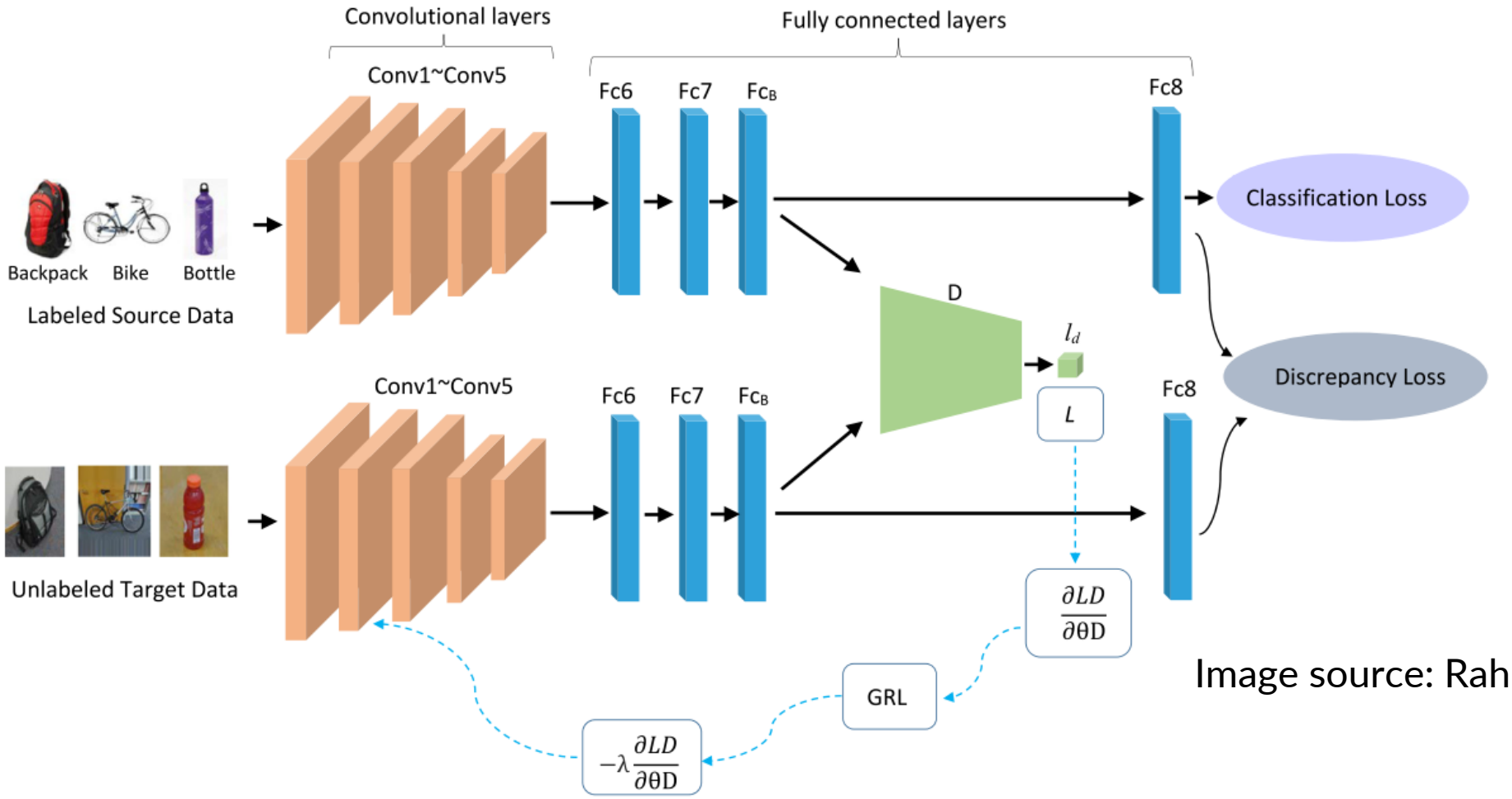


Image source: Rahman et al 2020

Challenges in repsesentation learning

Challenges in representation learning

- Interpretability of the features
  - Human's ability to predict what the features mean
  - Who takes the responsibility in critical applications (medical, security…)?
- Robustness in surprising real-world scenarios
  - Difficult to prepare for all the possible situations
- Vulnerability to adversarial attacks
- Sharing and availability
  - Huge datasets are needed to learn representations for general purposes
  - Huge computational resources are needed to train models
  - Sharing models can save a lot of time and resources

References

References

- Baghaei, Kourosh T., et al. "Deep Representation Learning: Fundamentals, Technologies, Applications, and Open Challenges." *IEEE Access* (2023).
- Bengio, Yoshua, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013): 1798-1828.
- Brachmann, Anselm, Erhardt Barth, and Christoph Redies. "Using CNN features to better understand what makes visual artworks special." Frontiers in psychology 8 (2017): 266181.
- Ganin, Yaroslav, et al. "Domain-adversarial training of neural networks." Journal of machine learning research 17.59 (2016): 1-35.
- Guan, Hao, and Mingxia Liu. "DomainATM: domain adaptation toolbox for medical data analysis." NeuroImage 268 (2023): 119863.
- Kim, Kang-Jae, and Tae-Jin Eom. "Classification of papers using IR and NIR spectra and principal component analysis." *Journal of Korea Technical Association of The Pulp and Paper Industry* 48.1 (2016): 34-42.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

References

- Mukhlif, Abdulrahman Abbas, Belal Al-Khateeb, and Mazin Abed Mohammed. "Incorporating a novel dual transfer learning approach for medical images." Sensors 23.2 (2023): 570.
- Ng, Dianwen, and Mengling Feng. "Medical Image Recognition: An Explanation and Hands-On Example of Convolutional Networks." *Leveraging Data Science for Global Health* (2020): 263-284.
- Pitas, Ioannis, Lecture on Domain Adaptation, 2018, https://www.i-aida.org/resources/domain-adaptation/
- Rahman, Mohammad Mahfujur, et al. "Correlation-aware adversarial domain adaptation and generalization." Pattern Recognition 100 (2020): 107124.
- Raitoharju, Jenni. "Convolutional neural networks." *Deep learning for robot perception and cognition*. Academic Press, 2022. 35-69.
- Rasmus, Antti, et al. "Semi-supervised learning with ladder networks." Advances in neural information processing systems 28 (2015).
- Yang, Xiangli, et al. "A survey on deep semi-supervised learning." IEEE Transactions on Knowledge and Data Engineering 35.9 (2022): 8934-8954.
- Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. Springer International Publishing, 2014.