Neural Networks 1 - Convolutional neural networks, Self-organization

18NES1 - Lecture 11, Summer semester 2024/25

Zuzana Petříčková

April 29th, 2025

What We Covered Last Time

Convolutional Neural Networks

- Classic CNN architecture + demonstration and visualizatuion how CNNs work - on MNIST data
- Efficient processing of image data with data loaders
- Training a CNN from scratch
- Techniques to improve generalization in CNNs
- Introduction to transfer learning

This Week

Convolutional Neural Networks – Completion

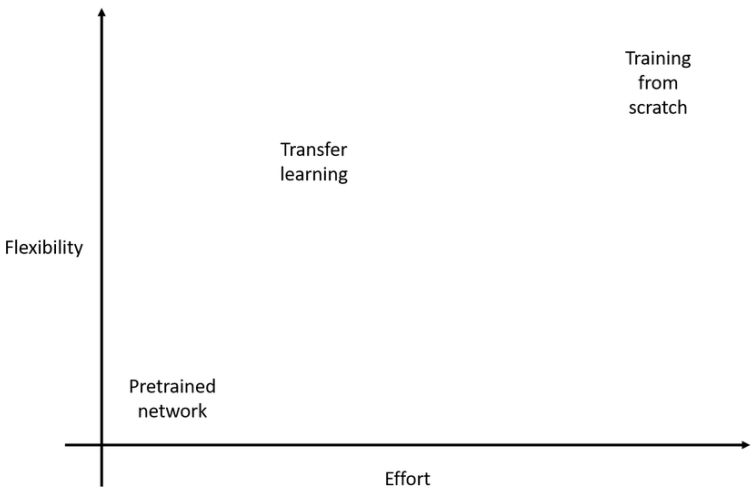
- Transfer learning and fine-tuning
- Overview of well-known convolutional neural network architectures
- Applications of convolutional neural networks to various types of tasks
- Brief introduction to other deep learning models
- Demonstrations and examples

Self-organization (Unsupervised Learning)

- Introduction

Ways to Build and Train a Convolutional Neural Network

- Training from scratch
- Using a pretrained model
- Transfer learning
- Fine-tuning a pretrained model



Source:

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

Reminder: Training a Model from Scratch on a Small Dataset – Flower Classification

- Dataset: **Oxford 102 Flower Dataset** paperswithcode.com/dataset/oxford-102-flower
- Contains **8189 images** across **102 classes** (each class has 40–258 images)
- Dataset size: approximately **330 MB** (JPEG format)
- Suitable for experimenting with both training CNNs from scratch and applying transfer learning

Tasks:

1. Classification into the 3 most populous classes
2. Classification into all 102 classes
3. For both tasks, we trained a basic bipyramidal CNN model

CNN_from_scratch_flowers_3.ipynb
CNN_from_scratch_flowers_102.ipynb

Reminder: Training a Model from Scratch on a Small Dataset – Flower Classification

CNN_from_scratch_flowers_3.ipynb

Observations

- Compared to MLPs, CNNs train relatively slowly (a GPU is highly recommended)
- **3 classes:** The basic model (without regularization) performed reasonably well, though slight overfitting occurred.
- **102 classes:** The model suffered significant overfitting.

Next Step

- Could a regularization technique help here?

Convolutional Neural Networks and Generalization

- Compared to MLPs, CNNs typically learn more slowly, especially on CPUs
- Convolutional neural networks tend to suffer more from overfitting
- Overfitting is a major issue when only a small dataset is available (hundreds or a few thousand samples)
- How can generalization be improved?
  - **Standard regularization**: early stopping, dropout, normalization (for deep models)
  - **Data augmentation**: expanding the dataset *on the fly*
  - **Transfer learning**

Data Augmentation

- Artificially increases the size and diversity of the training set
- Helps prevent overfitting and enhances model robustness by exposing it to a wider range of input variations
- Augmentation is performed **on the fly**, saving memory and increasing variability
- In Keras, implemented via layers such as `RandomFlip`, `RandomRotation`, `RandomZoom`, etc.
  - Various random image transformations (rotation, shifting, flipping, shearing, resizing, brightness and contrast adjustments, cropping, adding noise, blurring, combinations)

Source:

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

Reminder: Training a Model from Scratch on a Small Dataset – Flower Classification

CNN_from_scratch_flowers_3.ipynb

Observations

- **3 classes:** The basic model (without regularization) performed reasonably well, though slight overfitting occurred. A combination of techniques—dropout, early stopping, and data augmentation—helped achieve near-perfect training.
- **102 classes:** The model suffered significant overfitting. While regularization techniques (data augmentation + dropout + early stopping) greatly improved performance, the final accuracy remained just above 50%.

Next Step

- How about using a pretrained model or applying transfer learning?

Using a Pretrained Model

What if we could use an existing model trained on a large dataset?

Pretrained models:

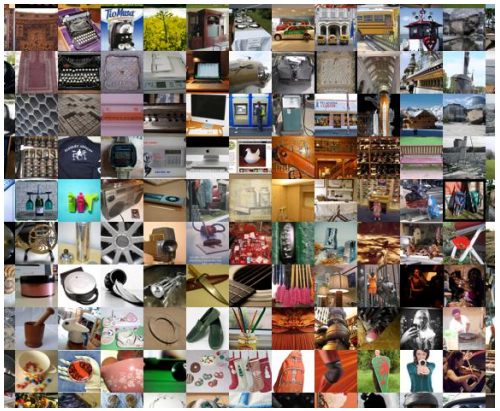
- https://keras.io/api/applications/
- Popular convolutional neural network architectures:
  - VGG16
  - MobileNet
  - ResNet
  - ...
- The model weights can be either randomly initialized or pretrained, typically on the **ImageNet** dataset

pretrained_model.ipynb

- A practical example of using a pretrained model

ImageNet Dataset

- 16 million color images from 20,000 categories
- Created as part of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC, 2010–2017)
- This challenge sparked the breakthrough of convolutional neural networks in image recognition
- ImageNet became the standard benchmark dataset for model comparison (replacing MNIST)



Source: https://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_full_1k.jpg

From Pretrained Models to Transfer Learning

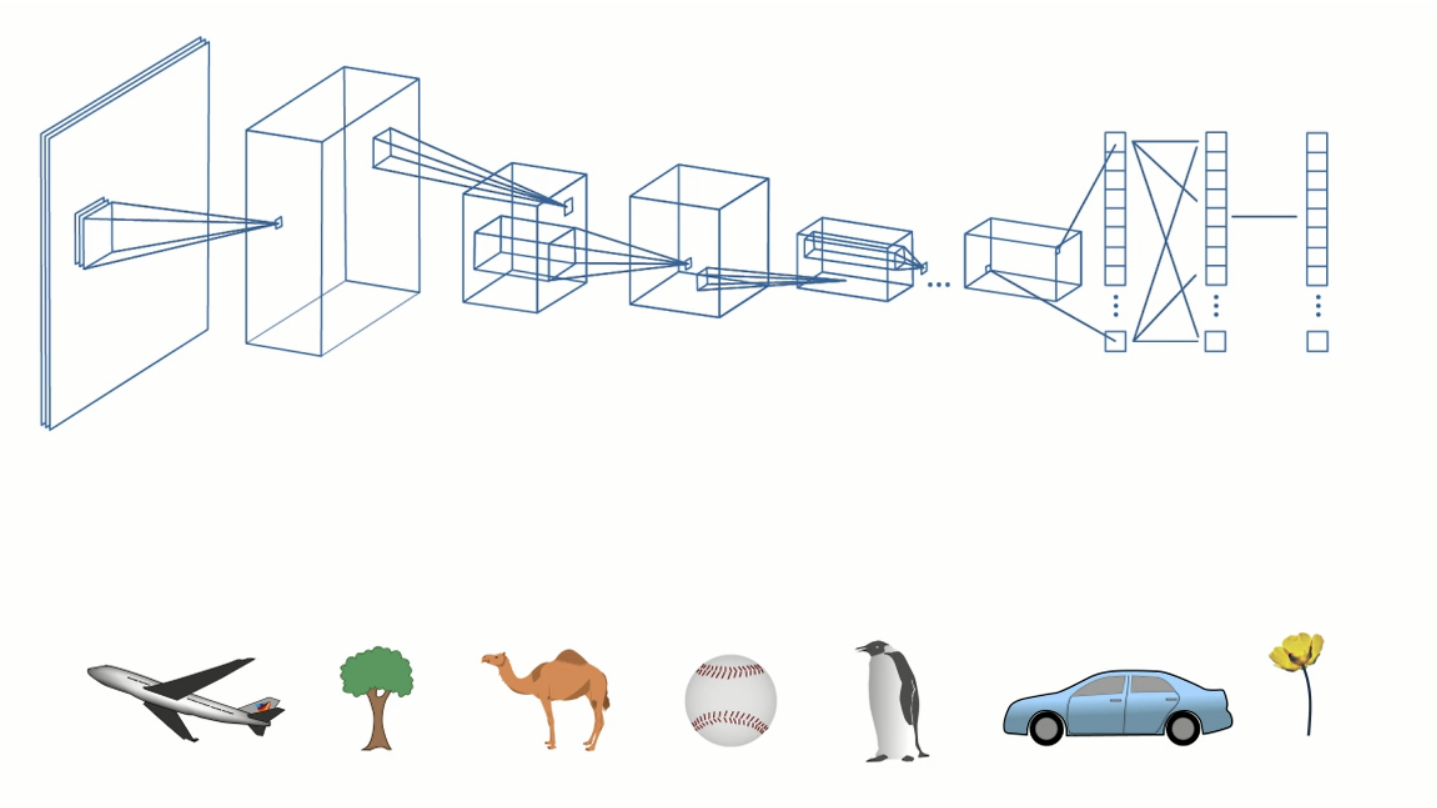
Using a pretrained model is a great starting point — but it usually won't be that simple.

- Although ImageNet contains 20,000 classes, the classification accuracy on your custom dataset may still be low.
- See our example: **pretrained_model.ipynb**

So how exactly do we adapt the model?

From a Pretrained Model to Transfer Learning

- Our pretrained model performs classification into 20,000 classes:

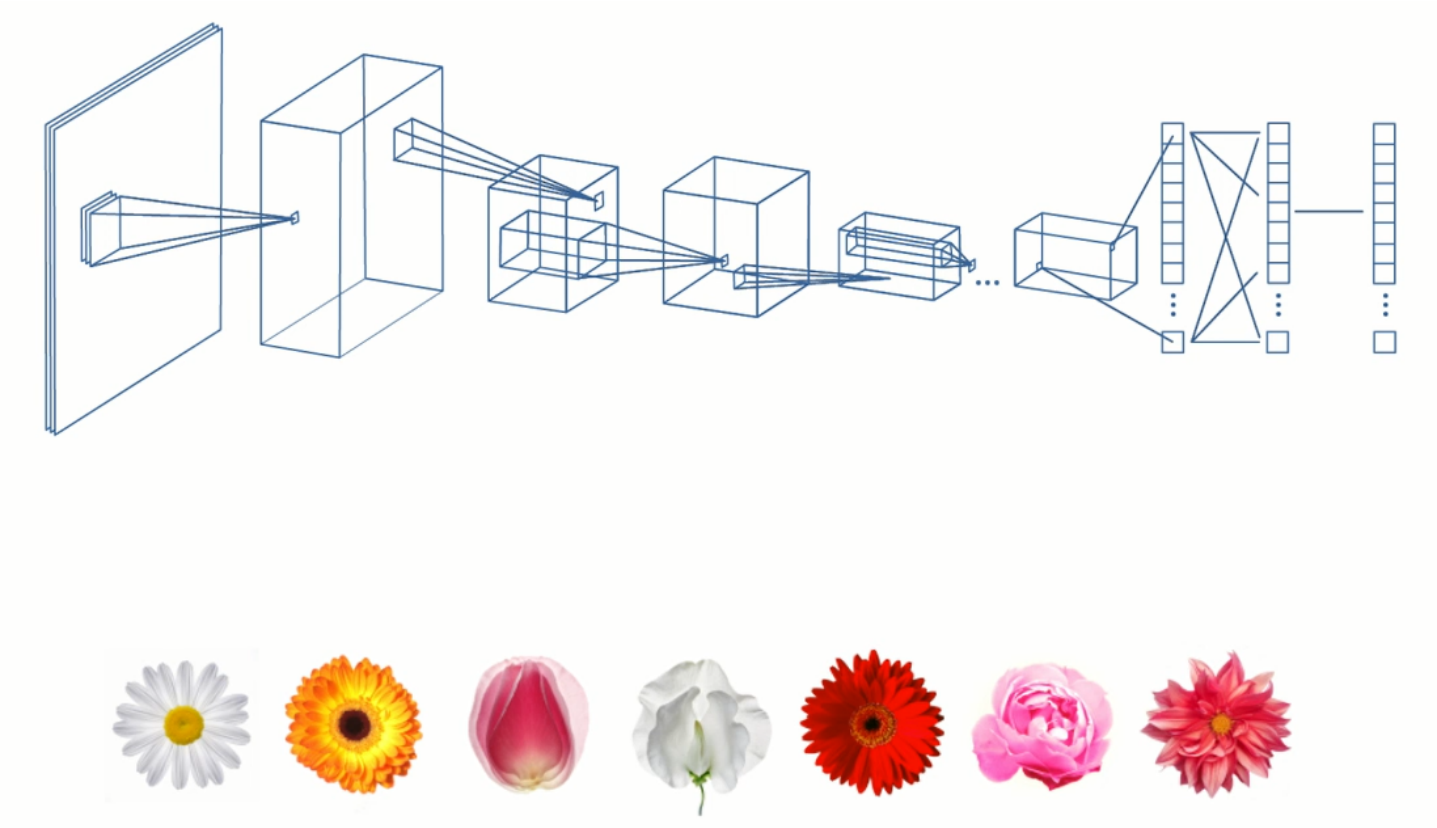


Source:

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

From a Pretrained Model to Transfer Learning

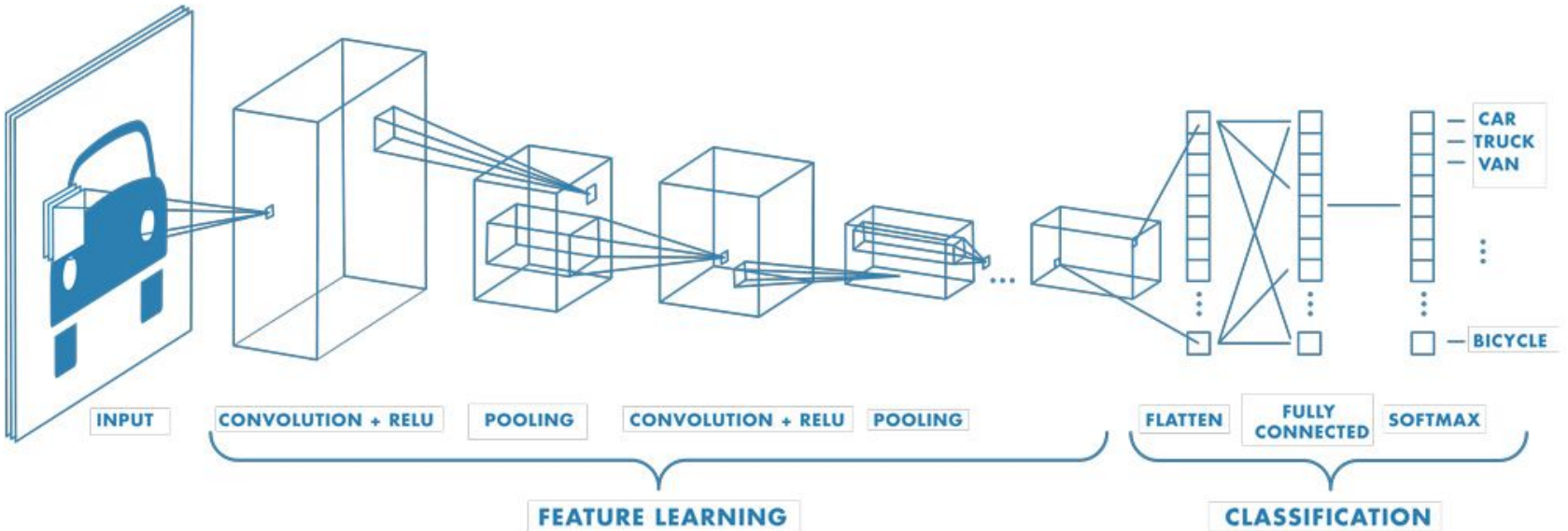
- But we need to classify into a different set of classes:



Source:

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

Recap: CNN Architecture



Components of a Convolutional Neural Network

- Convolutional base – extracts hierarchical features
- Flattening layer – converts data to a numeric vector
- Fully connected neural network for classification – **classification head**

Source:

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

From Pretrained Model to Transfer Learning

How exactly? First idea:

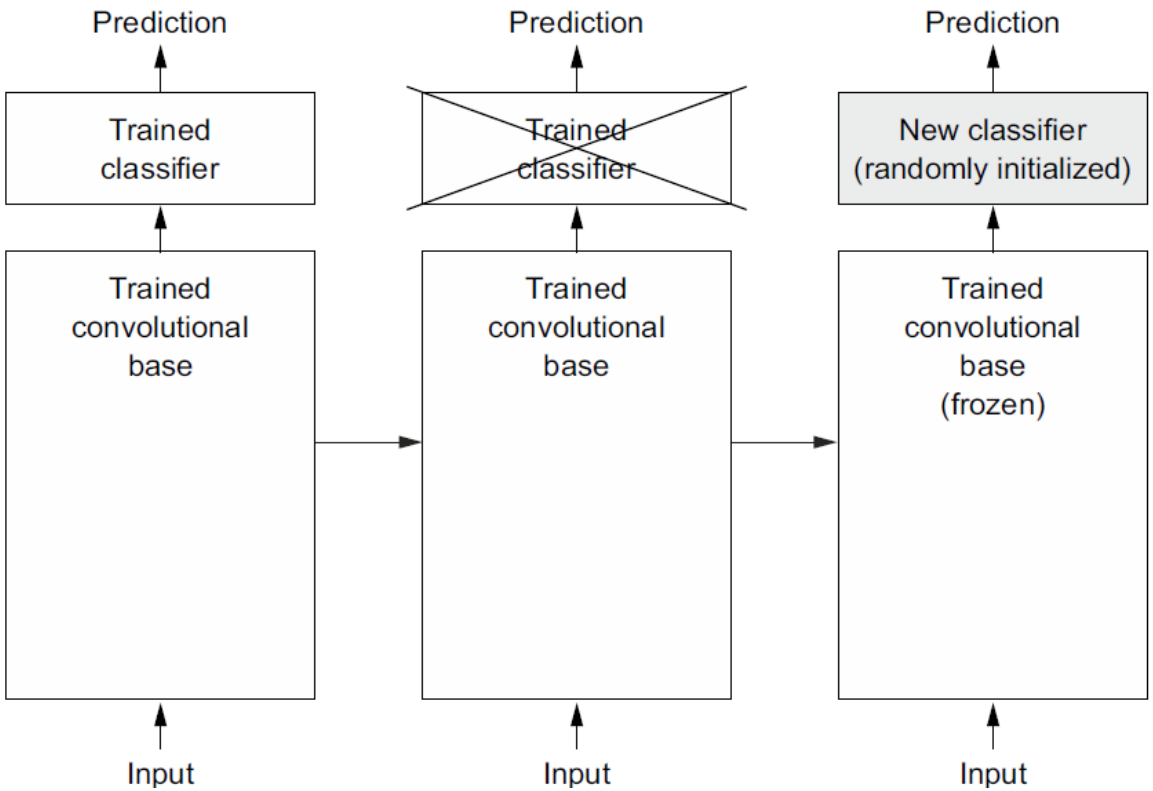
- Take a network pretrained on ImageNet
- Remove its classification head
- Use it to extract features from the new data $\rightarrow$ create a new training set
- Build a new fully connected neural network and train it on the extracted features

This approach is efficient, but often impractical:

- It's static – hard to apply built-in data augmentation tools

Transfer Learning

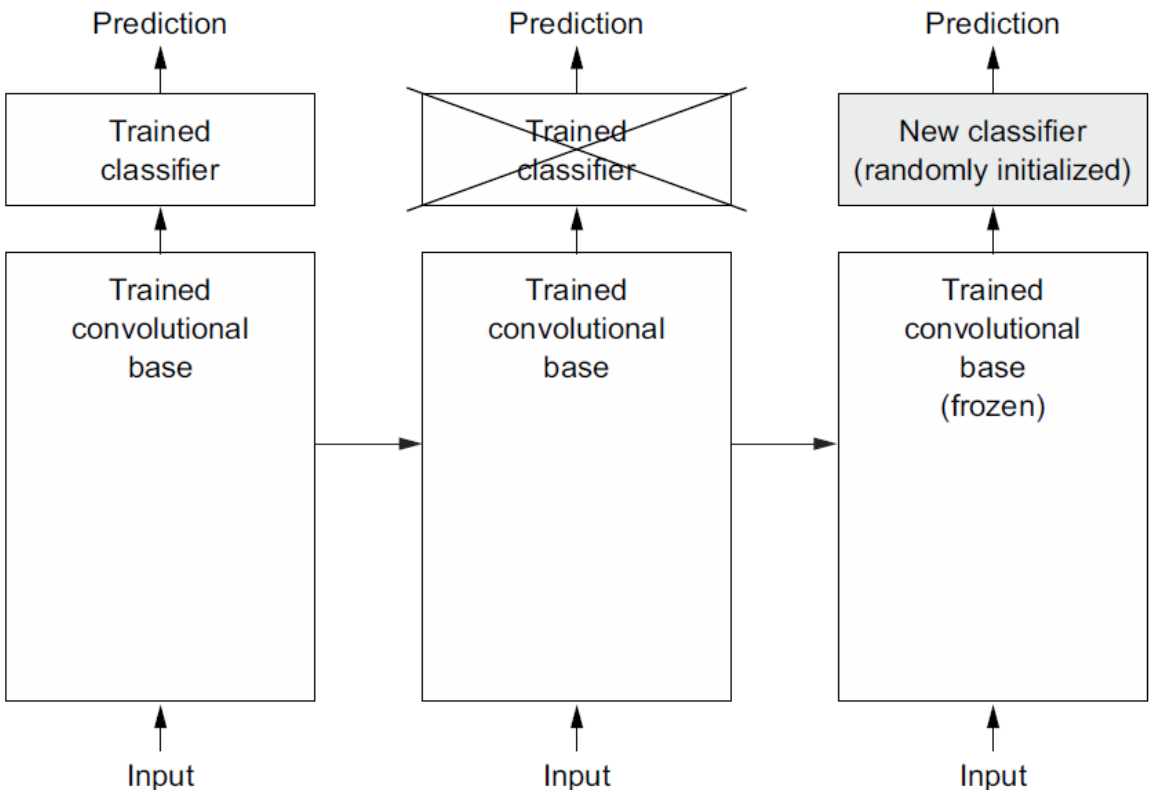
- Take a network pretrained on ImageNet
- Replace its classification head (or just its top part) with a new one (randomly initialized)



F. Chollet: Deep Learning with Python, Fig. 8.2

Transfer Learning

- Train the new classification head on your new data (keep the earlier layers frozen)



F. Chollet: Deep Learning with Python, Fig. 8.2

Transfer Learning – Practical Notes

- Typically use a smaller learning rate than when training from scratch
- Regularization (dropout, data augmentation) is useful

Examples:
CNN_transfer_learning_flowers_3.ipynb,
CNN_transfer_learning_flowers_102.ipynb

- Continuation of the Flowers 102 example

Transfer Learning

Practical Notes

- Typically, a smaller learning rate is chosen compared to training a network from scratch
- Regularization is recommended (dropout, data augmentation)

Examples: Flower Classification – Continued

- **CNN_transfer_learning_flowers_3.ipynb**
- **CNN_transfer_learning_flowers_102.ipynb**

Three Variants

1. Using a pretrained convolutional base for efficient feature extraction
2. Full transfer learning (combined with data augmentation and dropout)
3. Additional fine-tuning

Examples: Flower Classification – Continued

General Observations:

- **Feature extraction only:** Building the training set is time-consuming, but subsequent MLP training is very fast and results in excellent **accuracy**.
- **Full transfer learning:** Training is significantly slower (due to repeated inference through the convolutional base), but regularization techniques (data augmentation, dropout) further improve accuracy.

Task-specific Observations:

- **3 classes:** Both models train quickly in terms of epochs and achieve excellent final accuracy
- **102 classes:** Significant improvement compared to training from scratch, with accuracy exceeding 80%

Transfer Learning and Fine-Tuning

- A model trained via transfer learning can be further improved with fine-tuning:
  1. First, apply transfer learning (freezing weights outside the new classifier head)
  2. Then, unfreeze the deepest part of the convolutional base and continue training along with the new layers

Practical Notes

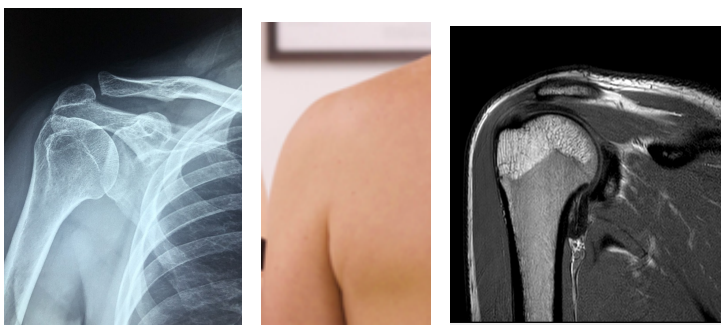
- Proceed carefully: start with a very small learning rate — ideally smaller than the final learning rate used in the pretrained model
- Regularization can also be applied

CNN_transfer_learning_flowers_102.ipynb

Limitations of Transfer Learning

- Transfer learning is suitable when training a model on data similar to the source domain (general model $\rightarrow$ more specific model)
- It cannot be applied if input layers need to be changed:
  - Different number of channels (e.g., X-ray images)
  - Different level of abstraction
  - Completely different types of data
- Sometimes, training a model from scratch is unavoidable

Example: Shoulder imaging extended with X-ray and MRI scans



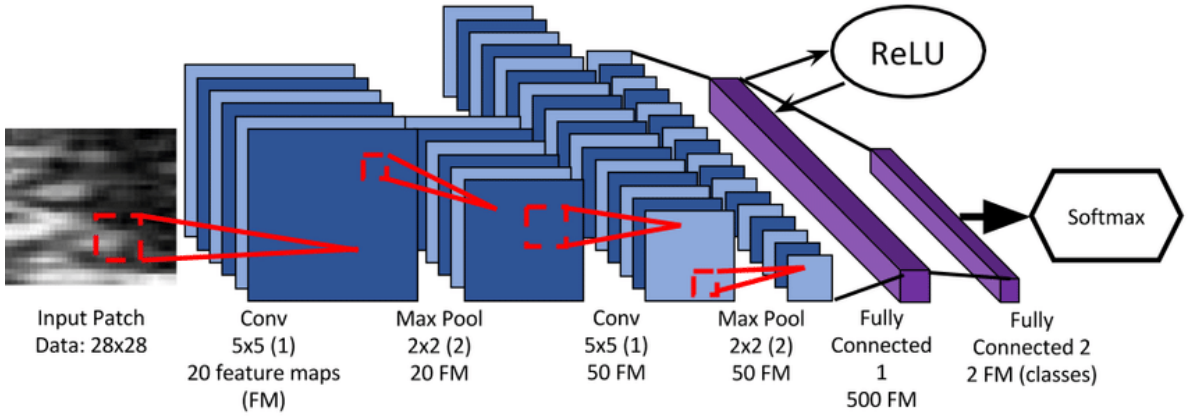
Source: `https://radiopaedia.org/cases/normal-shoulder-mri`

Well-Known Convolutional Neural Network Architectures

- The earliest architectures were wide and shallow, often with a deeper fully connected part, following a basic design scheme

LeNet-5

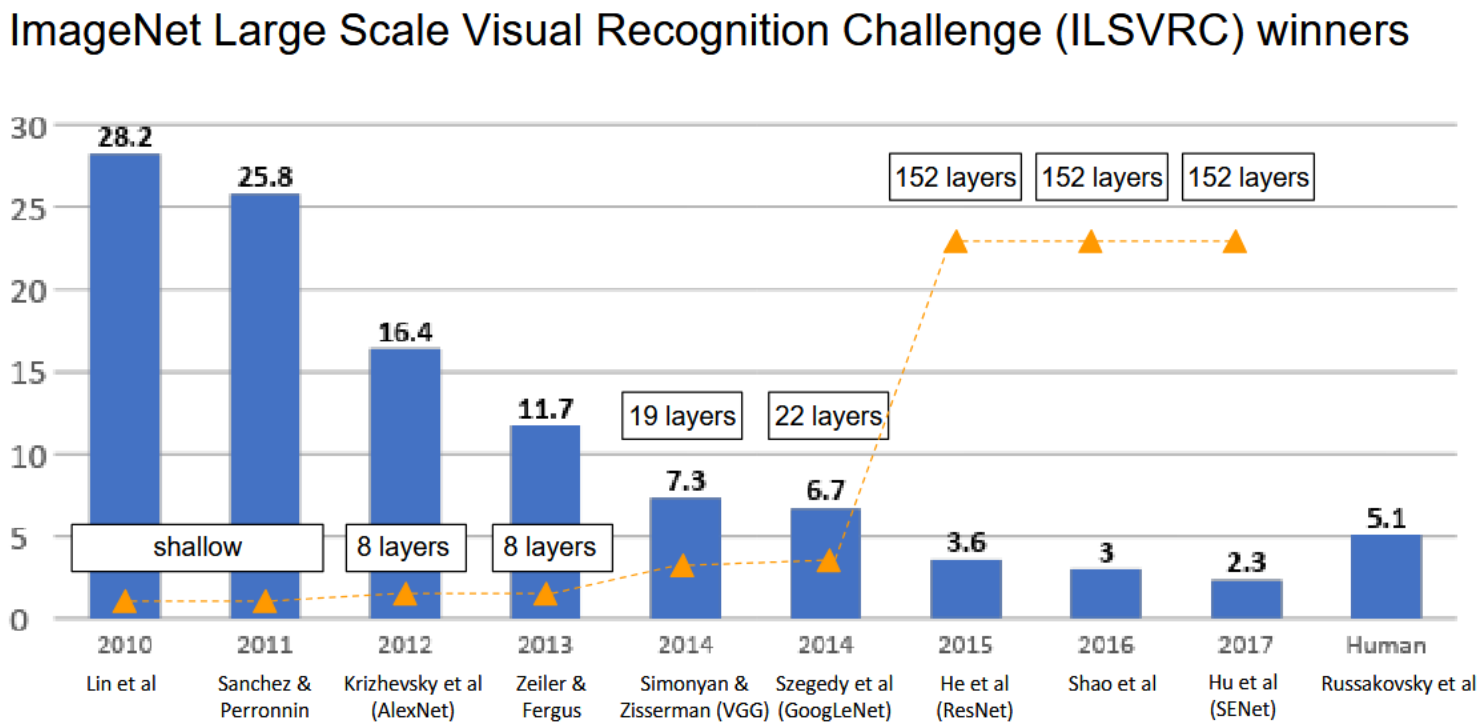
- One of the original architectures (Yann LeCun, 1998), relatively simple, trained on MNIST



Source of the image: M. H. Yap et al., "Automated Breast Ultrasound Lesions Detection Using Convolutional Neural Networks," IEEE Journal of Biomedical and Health Informatics, vol. 22, 2018

Well-Known Convolutional Neural Network Architectures

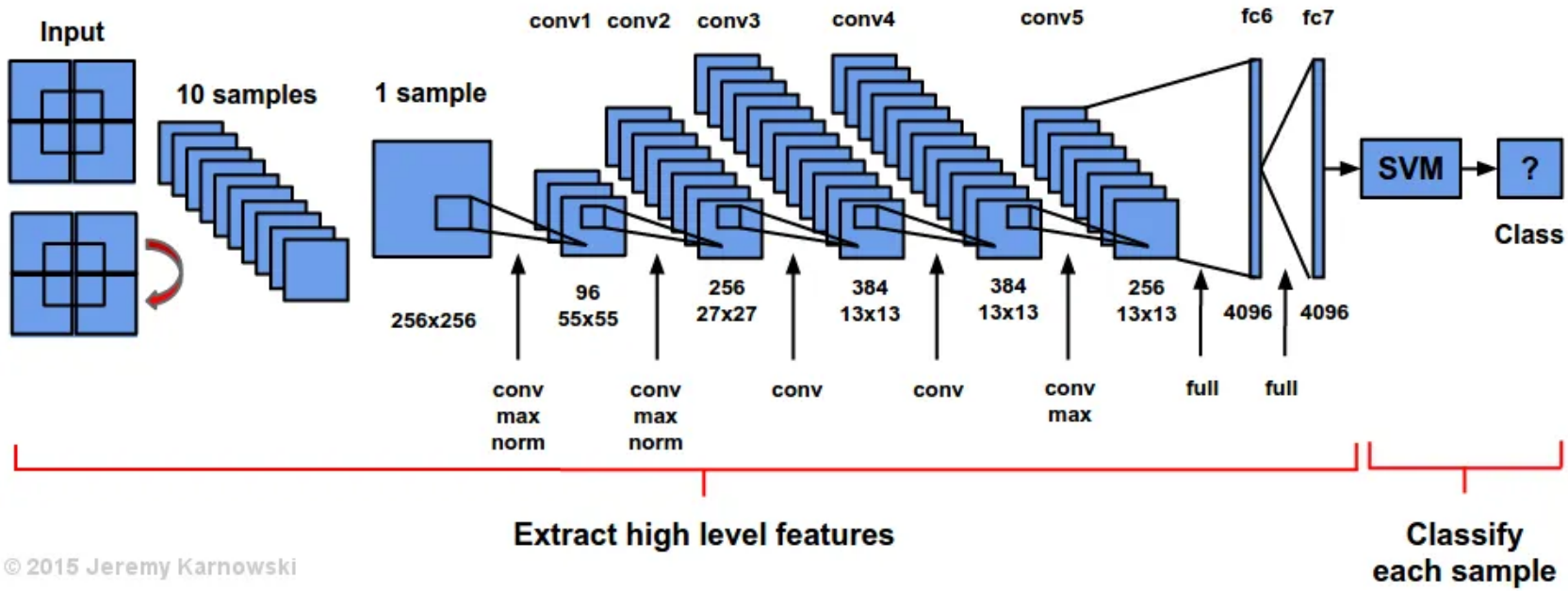
- The ILSVRC competition: a key contest using the ImageNet dataset



Source of the image: https://cs231n.stanford.edu/slides/2024/lecture_6_part_1.pdf

AlexNet

- The first CNN to win the ILSVRC competition (2012) on ImageNet — achieving 84.7% top-5 accuracy
- Significantly more complex: 8 layers, 61 million parameters, training took 5–6 days on 2 GPUs

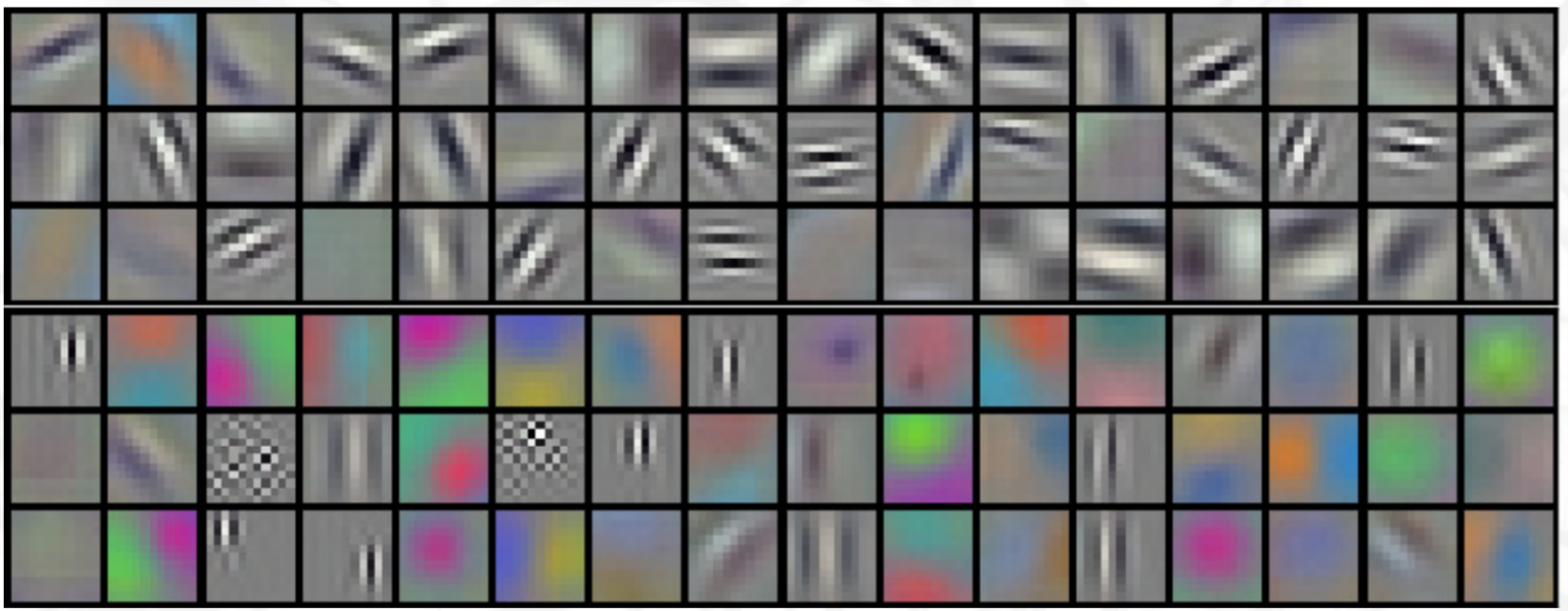


Source of the image:

https://medium.com/@jkarnows/alexnet-visualization-35577e5dcd1a

AlexNet – First Layer Filters

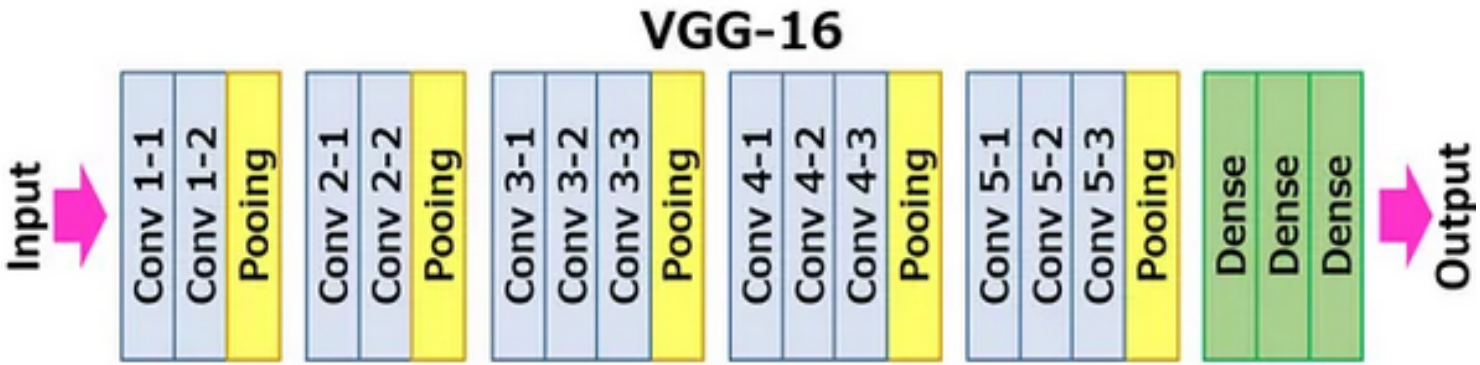
- 96 filters of size $11 \times 11 \times 3$ in the first convolutional layer of AlexNet

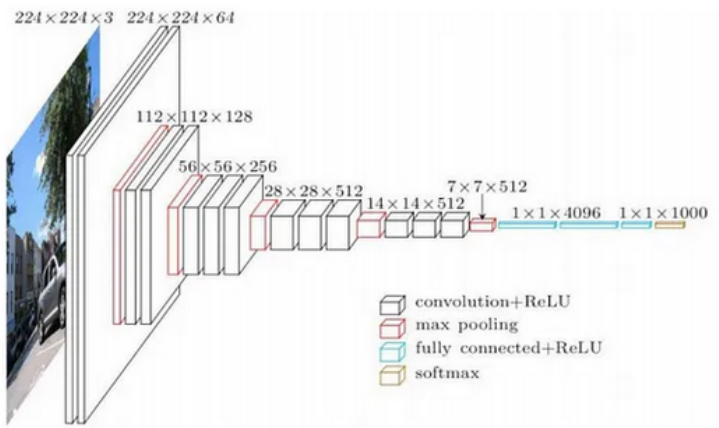


Source of the image: Alex Krizhevsky et al., "ImageNet Classification with Deep Convolutional Neural Networks", Figure 3

VGGNet

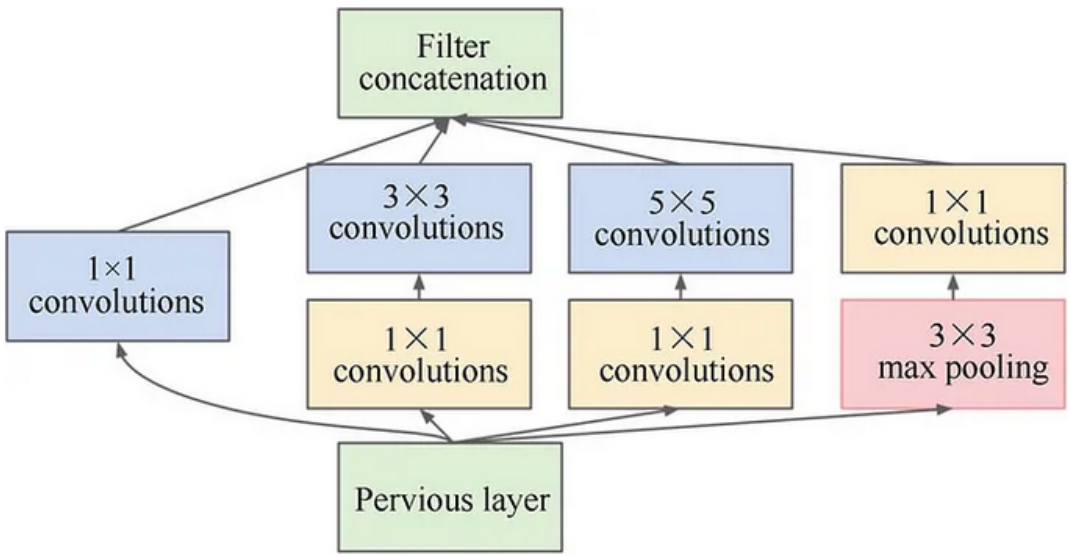
- Participant of ILSVRC 2014, a family of models (e.g., VGG16 – 16 layers, VGG19 – 19 layers) achieving 92.7% top-5 accuracy
- Pyramid-shaped architecture





GoogLeNet (Inception v1), 2014

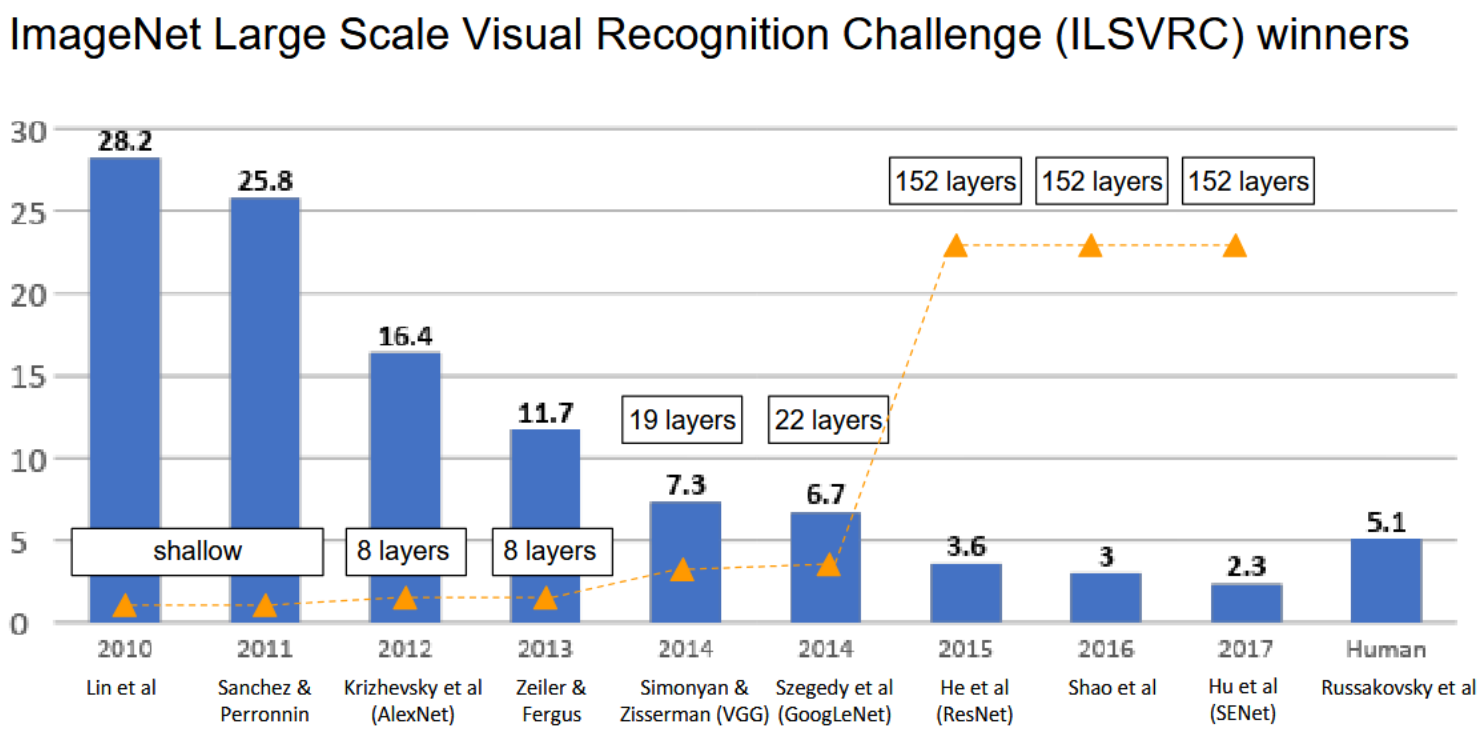
- Winner of ILSVRC 2014, 22 layers — 93.33% top-5 accuracy
- Revolutionary element: Inception blocks — feature extraction at multiple scales, parallelism, efficiency



The illustration is from the original paper[1]

Well-Known Convolutional Neural Network Architectures

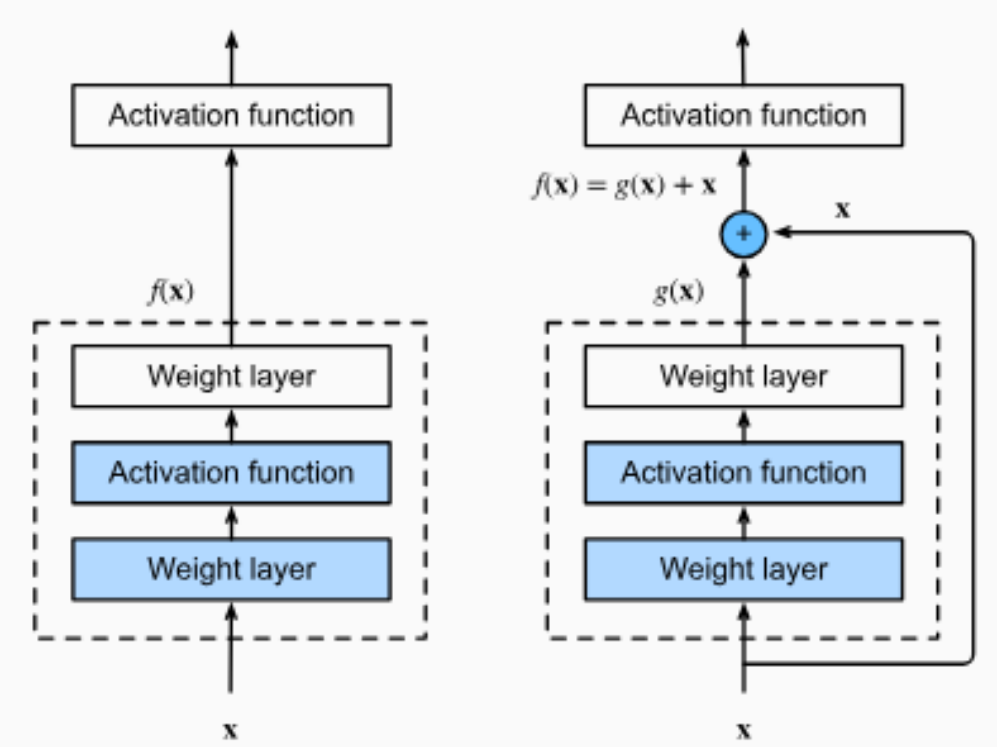
- After 2014, ILSVRC competition: significantly deeper and narrower architectures (reducing redundancy: depthwise separable convolutions), modular design



Source: https://cs231n.stanford.edu/slides/2024/lecture_6_part_1.pdf

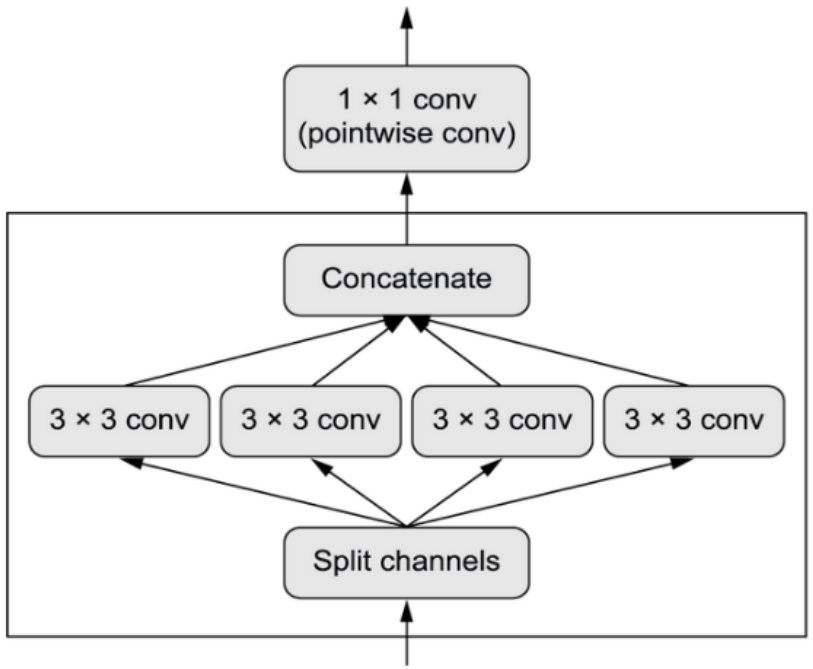
ResNet, 2015

- Winner of ILSVRC 2015, 152 layers — 96.4% top-5 accuracy
- Revolutionary element: skip connections — solving the vanishing gradient problem
- Residual blocks:



Depthwise Separable Convolutions (Xception, 2017)

- Another breakthrough to improve efficiency and slim down layers
- Spatial convolutions applied independently per channel, followed by channel mixing via pointwise convolution



Source: F. Chollet, "Deep Learning with Python," Figure 9.10

Other Popular Architectures

- Inception v2, v3 (2015, 2016)
- DenseNet (Gao Huang, 2016)
- MobileNet (Google, 2017), EfficientNet (2019), SqueezeNet (2016) — focused on reducing computational cost
- NASNet (Neural Architecture Search Network, 2017) — model architecture learned automatically via genetic algorithms and reinforcement learning
- ConvNeXt, 2020
- ...

In Keras:

- https://keras.io/api/applications/

Other Popular Benchmark Datasets

CIFAR-10, CIFAR-100

https://www.cs.toronto.edu/ kriz/cifar.html

- CIFAR-10: 60,000 color images, 10 classes (cars, dogs, . . .)
- CIFAR-100: 100 classes
- Small images, dataset popular for quick model testing and benchmarking

COCO (Common Objects in Context)

- Designed for object detection, segmentation, and image captioning

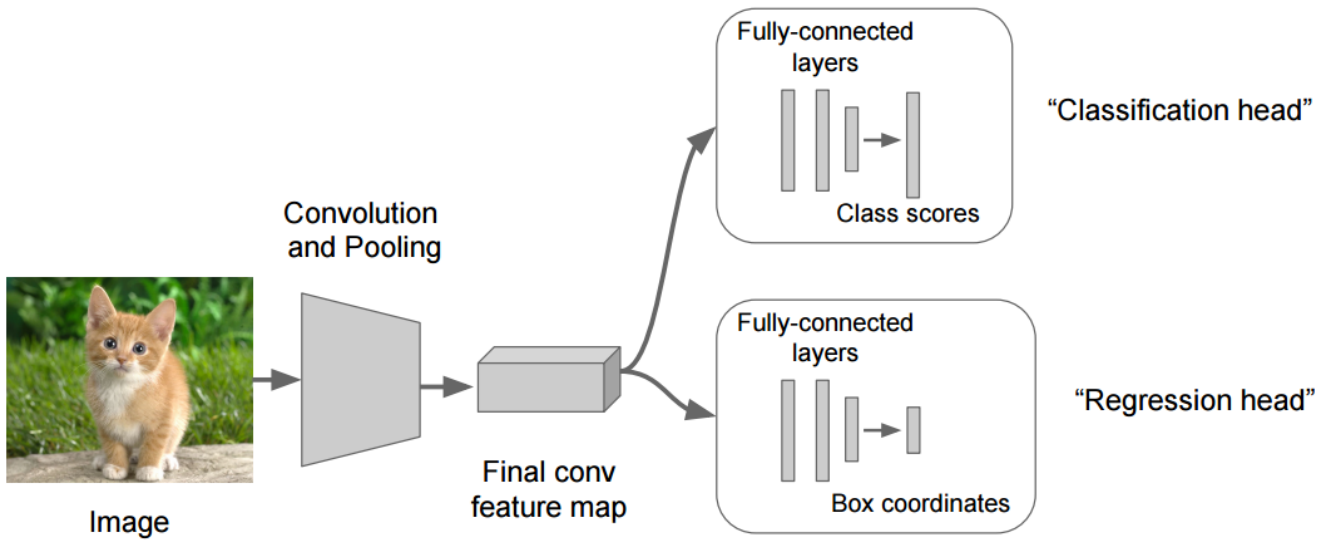
Source: `https://cocodataset.org/`

Evolution of CNN Architectures: Summary

- **Early CNNs** (e.g., LeNet-5, 1998): shallow and wide, simple fully connected parts
- **AlexNet** (2012): deeper networks, use of GPUs, ReLU activation, data augmentation, dropout
- **VGGNet** (2014): deeper and structured (pyramidal) networks, small convolutional filters (3×3)
- **GoogLeNet / Inception** (2014): multi-scale feature extraction, parallelism, improved efficiency
- **ResNet** (2015): extremely deep networks enabled by skip connections (residual learning)
- **Xception, MobileNet, EfficientNet** (2017–2019): model slimming via depthwise separable convolutions and architecture optimization
- **NASNet and beyond**: automated design of optimal architectures (neural architecture search)
- **Current Focus:** Efficiency, scalability, and adaptability

Applications of Convolutional Neural Networks

- The classification head of a neural network can be replaced by a different head to solve a different task on the same (or similar) data
  - Different classification tasks — classification head
  - Regression tasks — regression head
  - ...

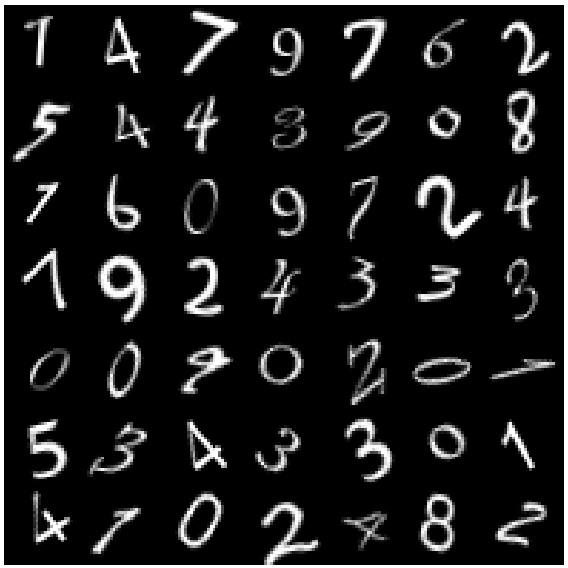


Source: https://i.stack.imgur.com/FGrD1.png

Applications of Convolutional Neural Networks

- Image classification — classification head
- Regression tasks — regression head

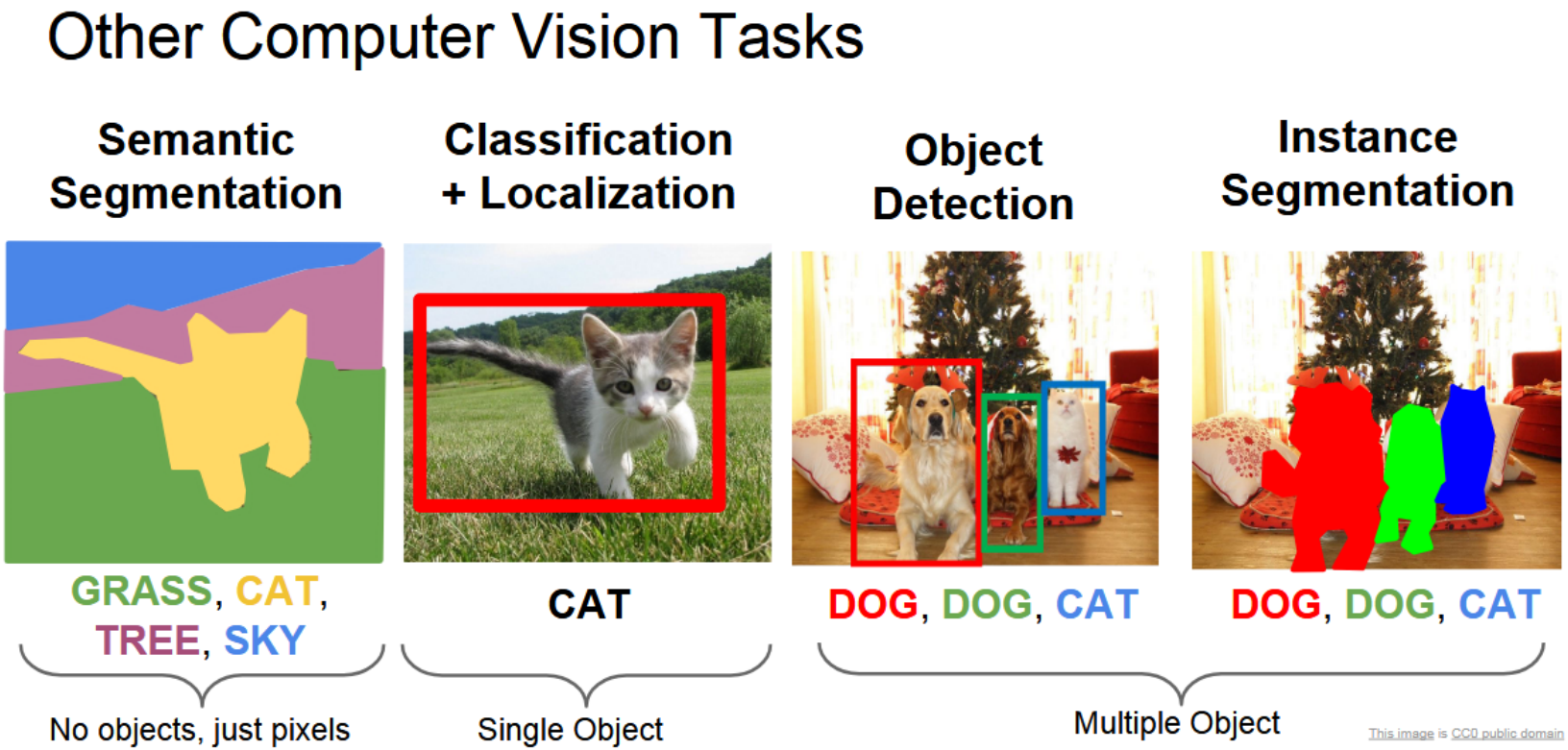
Example of Regression: Predicting the Angle of Digits



Source: https://matlabacademy.mathworks.com

Applications of Convolutional Neural Networks

- Image segmentation (semantic and instance segmentation), object detection

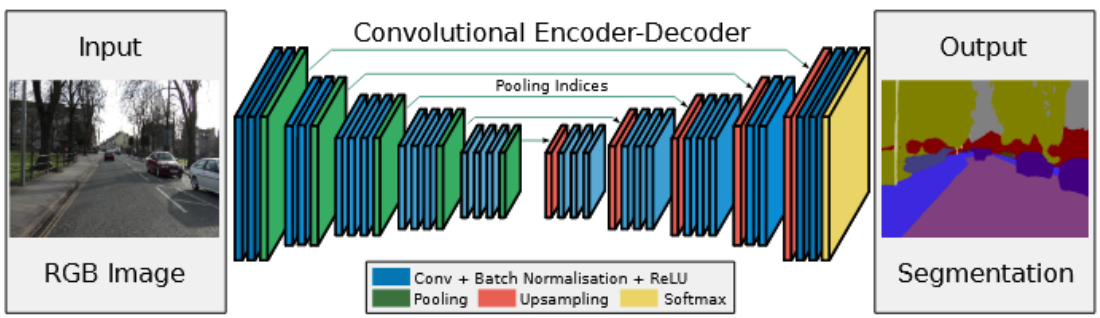


Source: `https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf`

Applications of Convolutional Neural Networks

Semantic Segmentation (SegNet, U-Net, . . .)

- Encoder-decoder architecture
- Encoder: classic CNN, feature extraction
- Latent space: compact representation
- Decoder: reconstructs the image to the original resolution



Source: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, `https://arxiv.org/pdf/1511.00561`

Applications of Convolutional Neural Networks

Object Detection: e.g., R-CNN (Region-based Convolutional Neural Network)

1. The image is divided into different regions (ROIs)
2. Each region is resized to a fixed size and passed through a pretrained CNN (e.g., VGG-16 trained on ImageNet)
3. Two heads are used: a classifier (SVM) to classify each region, and a regressor to refine the region boundaries

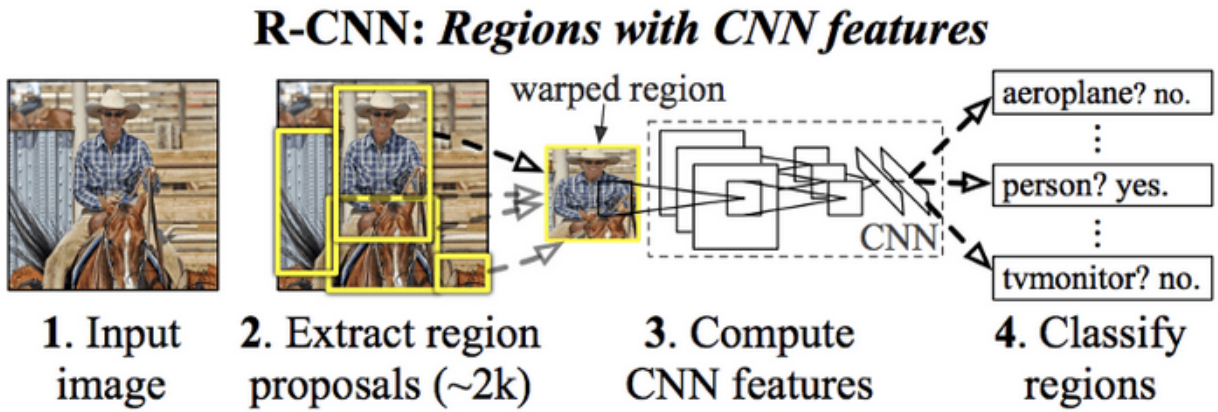
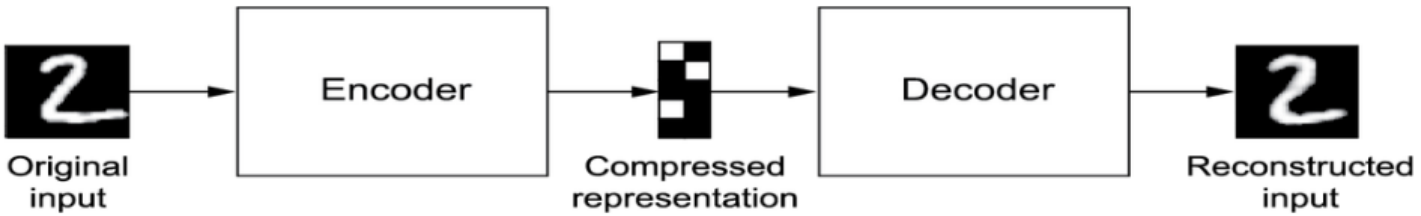


Image Source: R-CNN paper by Ross Girshick et al

Applications of Convolutional Neural Networks

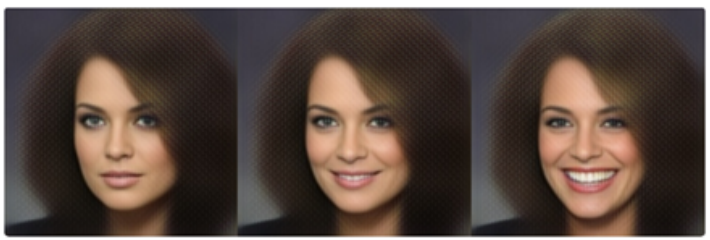
Autoencoders



F. Chollet: Deep Learning with Python, Fig. 12.4

Variational Autoencoders

- Encode data as a probability distribution
- Capable of generating new data samples or creating interpolations



F. Chollet: Deep learning v jazyku Python, obr. 5.7

Source: Tom White, "Sampling Generative

Applications of Convolutional Neural Networks

Processing 2D Images

- Image classification — classification head
- Regression tasks — regression head
- Object detection — detection head
- Image segmentation — segmentation head (encoder-decoder architecture)
- Image restoration, style transfer, image generation — autoencoder architecture
- Image captioning, pose estimation, facial feature detection, image similarity evaluation, ...

Other Data Types

- Video analysis (3D convolutions) — action recognition (e.g., in sports recordings)
- Sequential data (1D convolutions) — time series, audio data, limited use for natural language

Advantages and Disadvantages of Convolutional Neural Networks

- Well-suited for grid-like data (e.g., images)
- Invariance to translation, scale, and color changes
- Robust to noise in the data
- Computationally intensive training; requires large datasets and GPUs
- Risk of overfitting, especially with small datasets
- Vulnerable to adversarial examples — still an open problem *(small invisible perturbations causing incorrect classifications)* https://www.tensorflow.org/tutorials/generative/adversarial_fgsm

Other Deep Learning Models

Recurrent Neural Networks (RNNs)

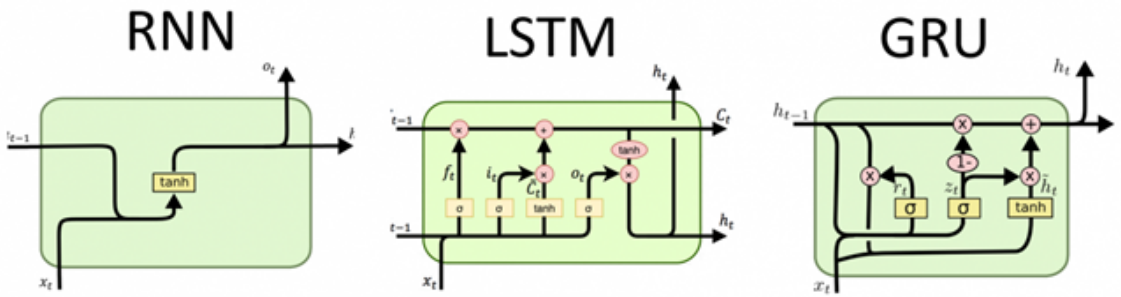
- Analysis and modeling of sequential data (time series, one-dimensional signals, speech, text, handwriting)
- Speech and handwriting recognition, machine translation

Long Short-Term Memory Networks (LSTMs)

- Models capable of capturing long-term dependencies in data

Gated Recurrent Units (GRUs)

- Simplified version of LSTM networks

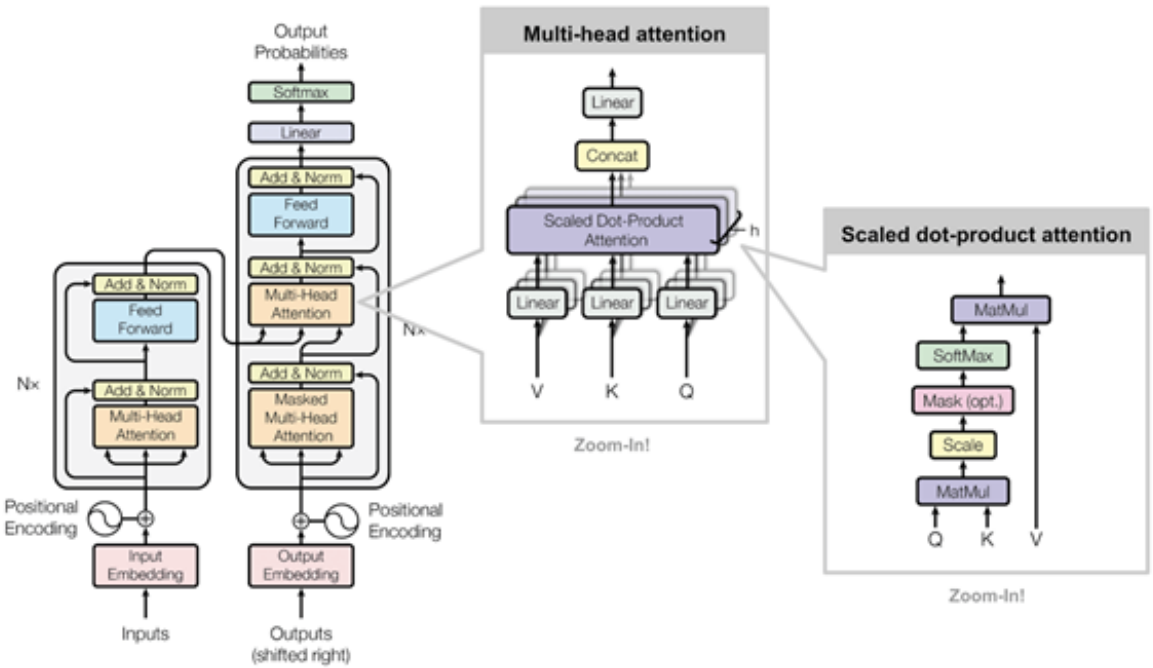


Source: https://cs231n.stanford.edu/slides/2020/lecture_10.pdf

Other Deep Learning Models

Transformer Networks (e.g., Gemini, GPT)

- Designed for natural language processing tasks (text classification, translation, text generation, etc.)
- Maintain context using the self-attention mechanism



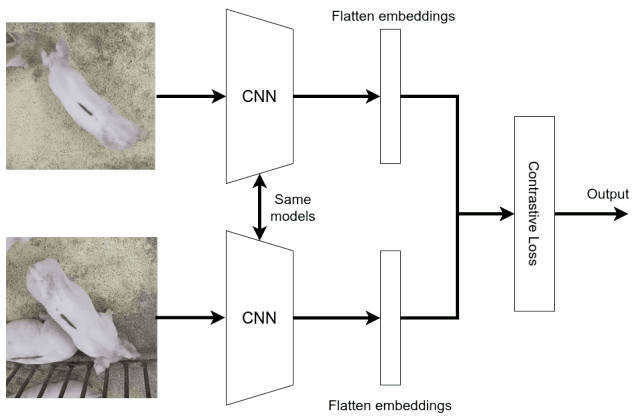
Source: A. Vaswani et al., "Attention is All You Need," Advances in Neural Information Processing Systems, 2017

Other Deep Learning Models

- Often modular architectures

Siamese Networks

- Used for image recognition and object tracking tasks
- Measure similarity between two different inputs
- **Example:** Animal recognition with Siamese Networks and Mean Embeddings

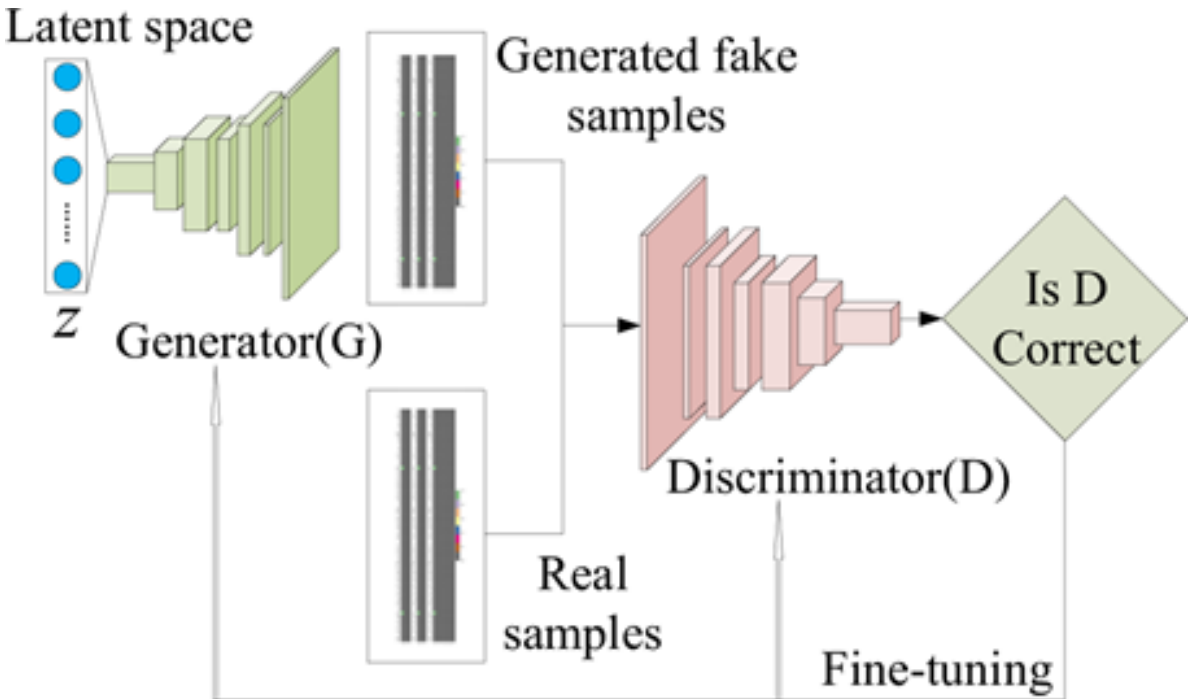


Source: `https://erdem.pl/2021/02/animal-recognition-with-siamese-networks-and-mean-embeddings`

Other Deep Learning Models

Generative Adversarial Networks (GANs)

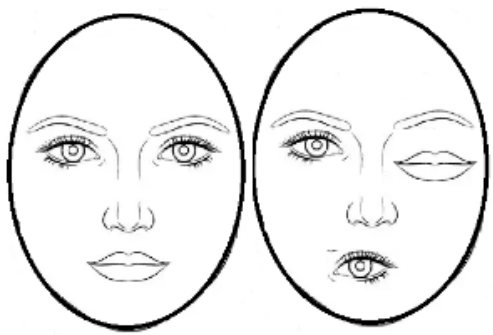
- Generate new data based on learned patterns



Source: Dan, Y., Zhao, Y., Li, X., et al., "Generative adversarial networks (GAN) based efficient sampling of chemical composition space for inverse design of inorganic materials," npj Computational Materials, 2020

Other Deep Learning Models

- **Deep Belief Networks (DBNs)**
  - Generative models trained in an unsupervised manner
- **Deep Q-Networks (DQNs)**
  - Used for reinforcement learning tasks
- **Capsule Networks**
  - Designed for image recognition tasks
  - Model hierarchical relationships between parts of objects



Source: `https://ilkeraliozkan.com.tr/publication/sengul-new-2022/sengul-new-2022.pdf`