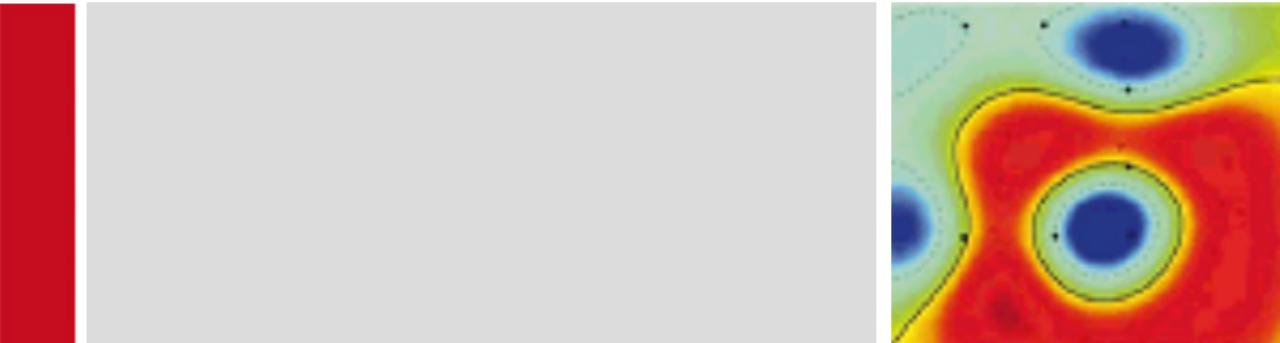




WiSe 2024/25

Deep Learning 1

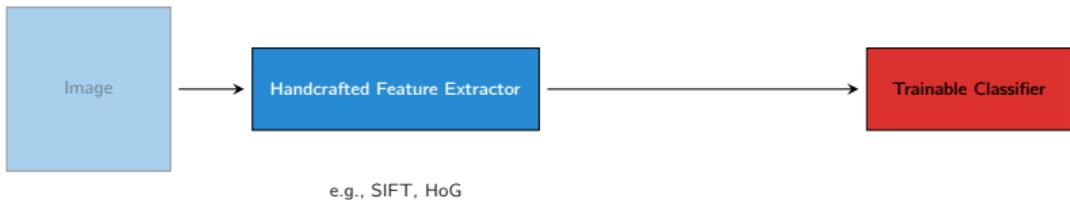


Lecture 8

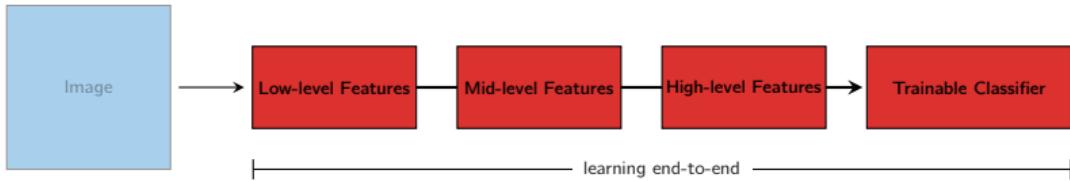
Convolution Neural Networks

Machine Learning for Computer Vision

- ▶ Traditional approach (before 2012): handcrafted features



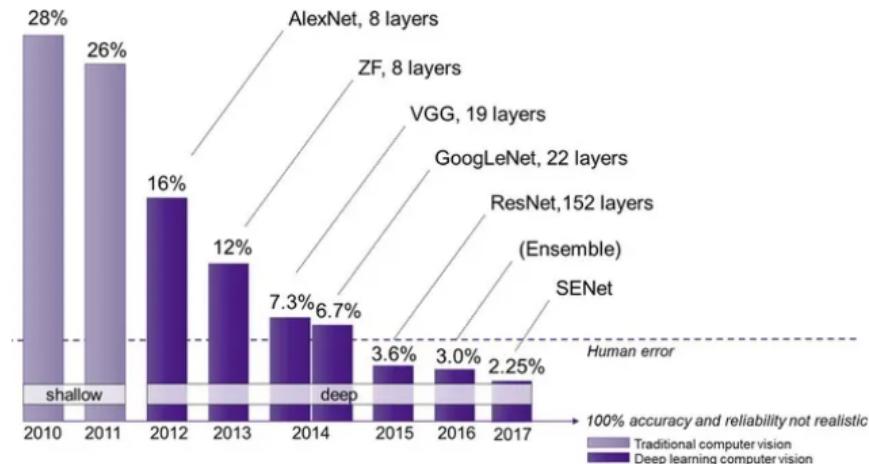
- ▶ Deep Learning (2012-...): end-to-end hierarchical feature learning



Figures are adapted from Canziani and LeCun, 2021.

ImageNet [2] Benchmark

Task: 1000-class classification (~3M images)



ImageNet Classification Top-5 Error (%). Gordon Cooper, 2019

Remarks: 1): Another key ingredient that helped facilitate the progress on the ImageNet task is the utilization of GPUs in training large CNNs [9]; 2): More up-to-date results see <https://paperswithcode.com/sota/image-classification-on-imagenet>.

Recap: Multi-layer perceptrons

A sequence of affine and thresholding transformations:

$$\boldsymbol{z}^{(1)} = W^{(1)} \boldsymbol{x} + \boldsymbol{b}^{(1)}$$

$$\boldsymbol{a}^{(1)} = \sigma(\boldsymbol{z}^{(1)})$$

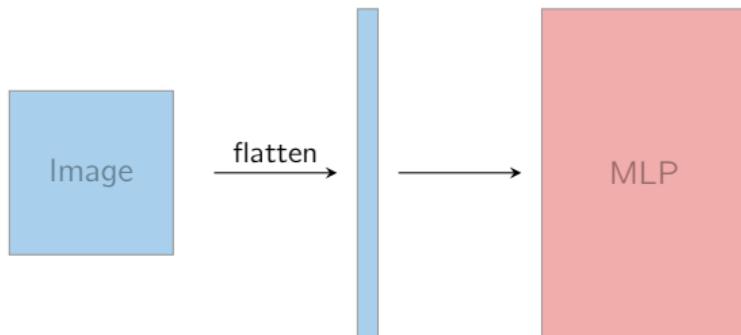
⋮

$$\boldsymbol{a}^{(j)} = \sigma(W^j \boldsymbol{a}^{(j-1)} + \boldsymbol{b}^{(j)})$$

⋮

Could we use MLPs for Images?

One might flatten images to be (very) tall vectors and feed these vectors as inputs to MLPs.



For example, a 3-color 32×32 image (e.g., $\mathbb{R}^{32 \times 32 \times 3}$) to a tall vector \mathbb{R}^{3072}

Issues

- ▶ space and algorithmic complexity
- ▶ statistical (learning) inefficiency: We do *not* exploit correlations of neighbouring pixels.

Could we use MLPs for Images? (cont.)

Issue 1: Space and Algorithmic Complexity Let m be the number of input dimensions (after flattened), e.g., $m = 3072$ and n be the number of neurons in the first layer of a MLP.

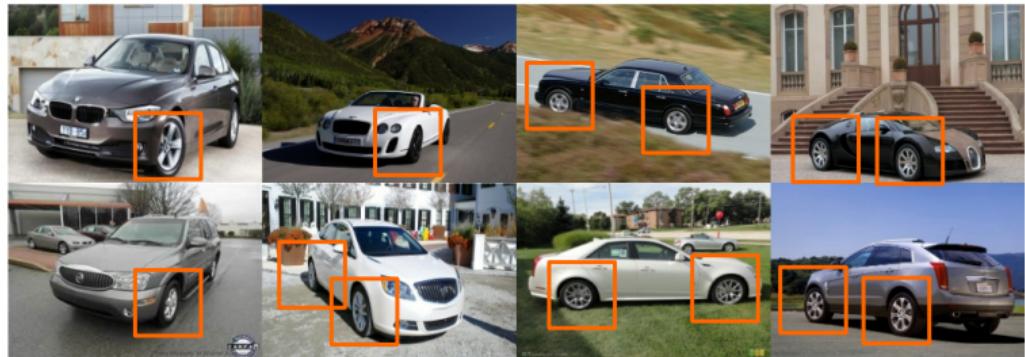
Space Complexity: We need $mn + n$ parameters.
→ mn for weights $W^{(1)}$ and n for biases $b^{(1)}$.

Algorithmic Complexity: $O(mn)$

Could we use MLPs for Images? (cont.)

Issue 2: Statistical Inefficiency

After flattening, we do not exploit local relationships between neighbouring pixels.



For example, pixels in the regions of car wheels often correlate, and the correlation might be useful for some learning tasks.

Figure is adapted from the Stanford Car dataset [12].

Overview of CNNs

CNNs = learning **hierarchical features** (from low to high level features) using **convolution** (learning correlation between neighbour pixels) and **pooling** layers (enlarging the size of neighbourhoods)¹

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)

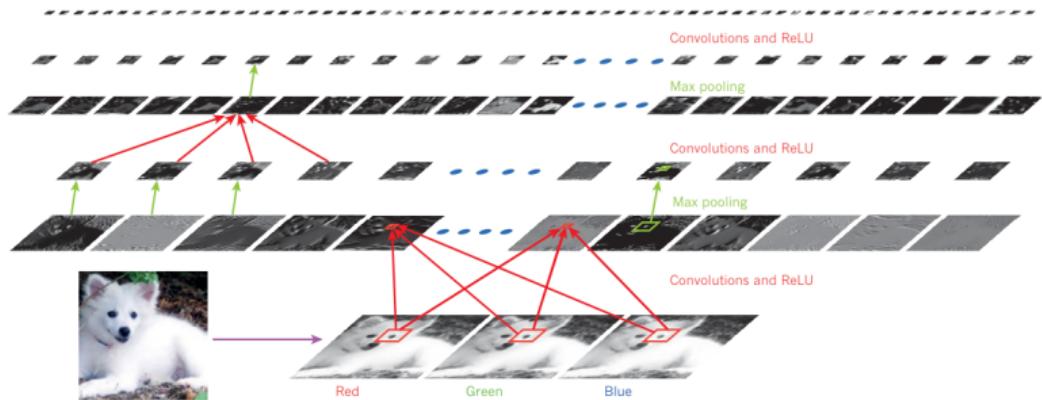


Figure 2 | Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map

corresponding to the output for one of the learned features, detected at each of the image positions. Information flows bottom up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output. ReLU, rectified linear unit.

Remarks: 1): In practice, there are other components in CNNs that help increase the performance of the models, but convolution and pooling layers are the two main important ingredients; 2) Figure is taken from LeCun et al., 2015 [15].

Convolution Operator

Let $x(t) \in \mathbb{R}$ be a one-dimensional signal at time $t \in \mathbb{R}$ and $w(\tau)$ be a weighting function. The convolution operator $*$ is

$$\begin{aligned}(x * w)(t) &= \int x(\tau)w(t - \tau) \, d\tau && \text{(continuous setting)} \\ &\approx \sum_{\tau=-\infty}^{\infty} x(\tau)w(t - \tau) && \text{(discrete setting)}\end{aligned}$$

We refer

- ▶ x as the **input**,
- ▶ w as the **kernel**,
- ▶ and $(x * w)(t)$ as **the feature map**, denoted with $a(t)$.

Convolution Operator (cont.)

We can see the convolution operator is **commutative**

$$\begin{aligned}(x * w)(t) &= \sum_{\tau} x(\tau)w(t - \tau) \\&= \sum_{\tau'} x(t - \tau')w(\tau') \quad (\text{define } \tau' := t - \tau) \\&= (w * x)(t)\end{aligned}$$

In deep learning, we instead use a related operator called **cross-correlation**¹, denoted with \star (NOT asterisk $*$)

$$(x \star w)(t) = \sum_{\tau'} x(t + \tau')w(\tau').$$

Remarks: 1): the operator is generally referred and the underlying implementation of convolution layers in deep learning frameworks; therefore, we shall strict with the name *convolution* and use the symbol to indicate the actual operator; 2) The range of τ' will be made it clear in the following slides.

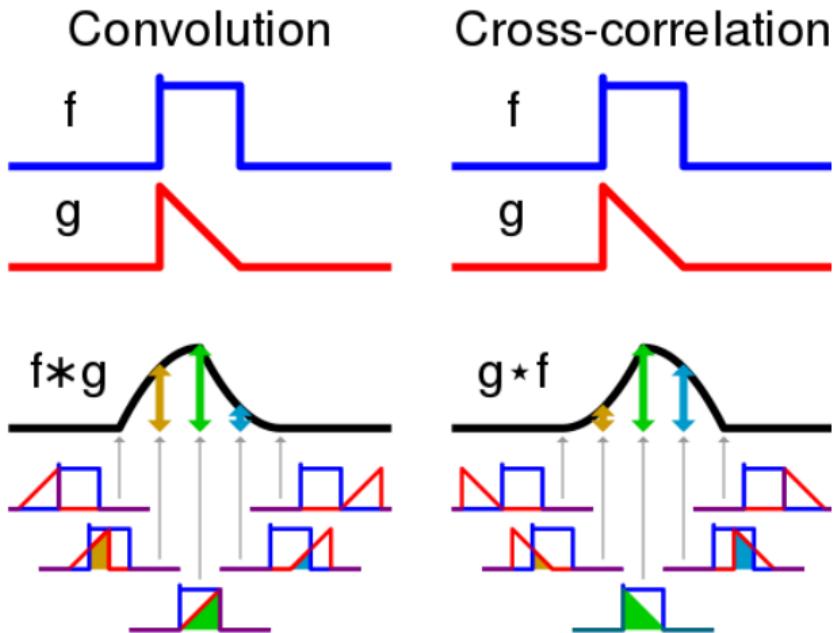
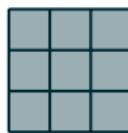


Figure from Wikipedia: Convolution.

Two-Dimensional Discrete Convolution

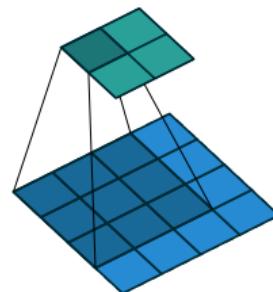
A two-dimensional discrete convolution layer consists of

- ▶ weight ($W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}} \times k \times k}$, or kernel) whose dimensions governed by
 - ▶ Number of input channels d_{in} , e.g., a 3-color image $d_{\text{in}} = 3$
 - ▶ Number of output channels d_{out} chosen by the user
 - ▶ Kernel size k chosen by the user
- ▶ bias ($b \in \mathbb{R}$, optional)



Weight W

$(k = 3, d_{\text{in}} = 1, d_{\text{out}} = 1)$



Discrete convolution using W (no bias) on an input ($d_{\text{in}} = 1$)

Figures are adapted from [3].

Numerical Example

0	1	2
2	2	0
0	1	2

Weight W ($k = 3$)

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

3	3	2	1	0
0	0	1	3	1
3	1	2_0	2_1	3_2
2	0	0_2	2_2	2_0
2	0	0_0	0_1	1_2

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

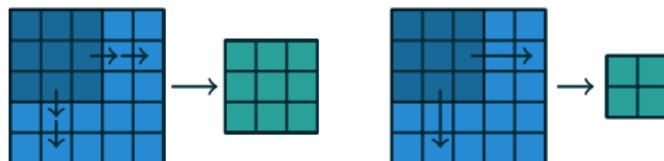
Left: Weight applied to two locations in the input;
Right: Output of the operator at the two locations
(shaded entries) and other locations.

Figures are from [3].

Stride and Padding

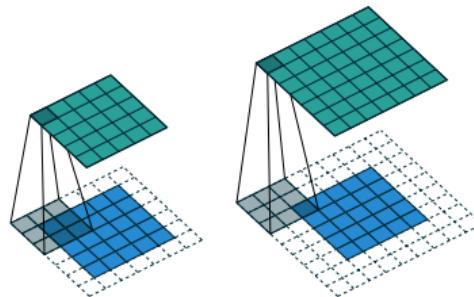
Apart from the parameters (W, b), discrete convolution also has two important hyper-parameters, namely

- ▶ Stride (amount of kernel translated)



Left: `stride = 1` (previous example), Right: `stride = 2`

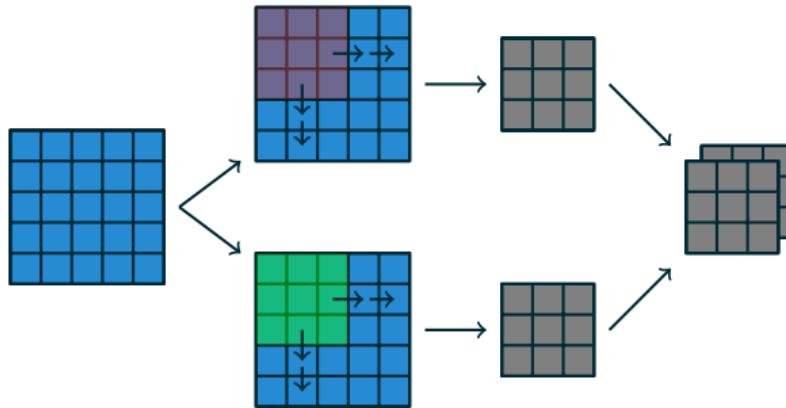
- ▶ Padding : How do we handle regions at the boundary of the input?



Padding with sizes 1 and 2 respectively.

Discrete Convolution when $d_{\text{out}} > 1$

So far, we mainly discuss the case when $d_{\text{out}} = 1$. When $d_{\text{out}} > 1$, we repeat the process d_{out} times and concatenate the output of each time together.



Example when $d_{\text{out}} = 2$ (**stride** = 1 and no padding)

Two-Dimensional Discrete Convolution

Denote $a \in \mathbb{R}^{d_{\text{in}} \times h_{\text{in}} \times w_{\text{in}}}$ be an input and $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}} \times k \times k}$ be a convolution weight. Two-dimensional discrete convolution can be expressed as

$$z_{c',m,n} = (a \star W)_{i,j} = \sum_{\tau_1=1}^k \sum_{\tau_2=1}^k \left[\sum_{c \in 1}^{d_{\text{in}}} a_{c,m+(\tau_1-1),n+(\tau_2-1)} W_{c',c,\tau_1,\tau_2} \right],$$

where $\forall c' \in \{1, \dots, d_{\text{out}}\}$, $\forall m \in \{1, \dots, h_{\text{out}}\}$, and $\forall n \in \{1, \dots, w_{\text{out}}\}$.

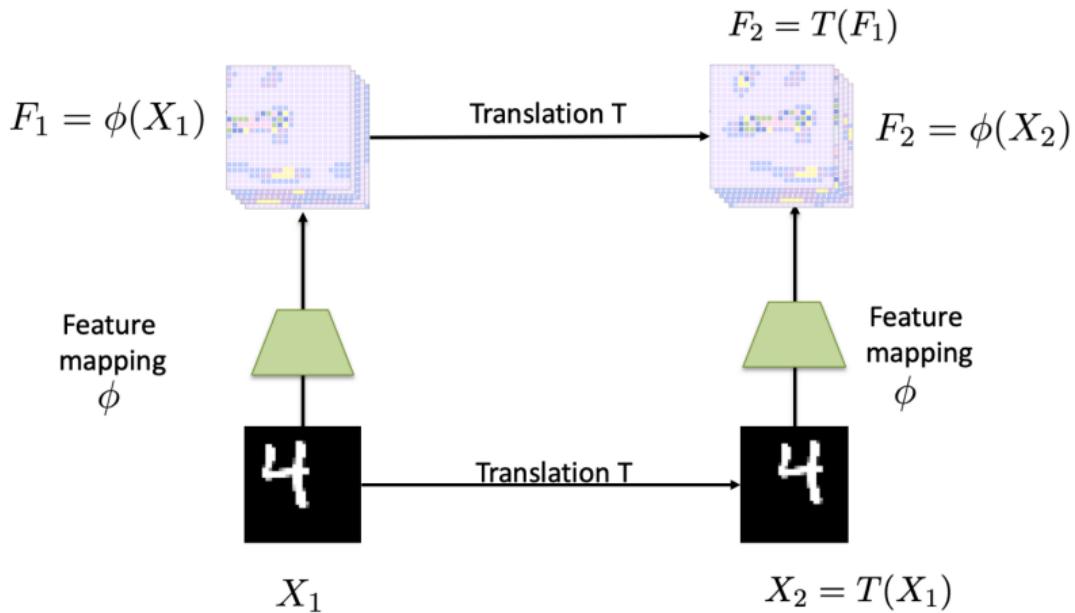
The exact value of h_{out} and w_{out} depends on the kernel size, stride, and padding.

Suppose we have a square input $h_{\text{in}} = w_{\text{in}}$ and using stride=1 and no padding. We have

$$h_{\text{out}} = w_{\text{out}} = h_{\text{in}} - k + 1 \quad (\text{Relationship 2 in [3]})$$

See [3] for the relationship between input and output size in other settings.

Translation Equivariance



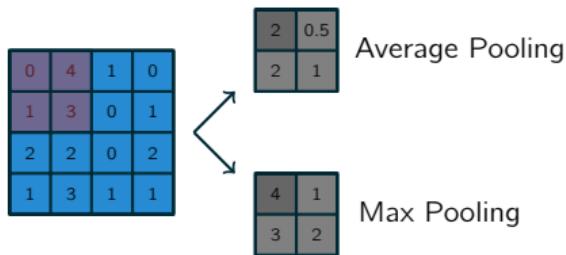
Credit: Christian Wolf

Pooling Layer

Convolution can detect patterns that are not larger than the kernel size k . A stack of convolutions can be used to increase this pattern-respond region, commonly referred to as **Receptive Field**.

Practically, it is more effective to increase the receptive field by **subsampling** the input, or **Pooling**.

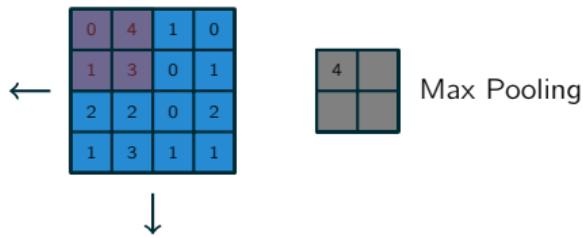
Commonly used pooling layers are average¹ and max pooling.



Average and max pooling with $k = \text{stride} = 2$.

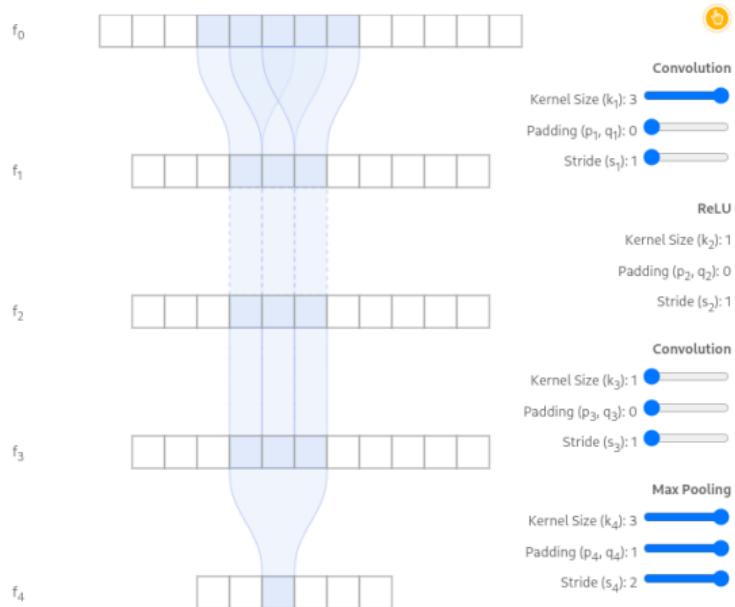
- 1): One can express the average pooling using convolution with constant weight $W_{\tau_1, \tau_2} = 1/k^2$ and no bias.

(Local) Translation Invariance of Max Pooling



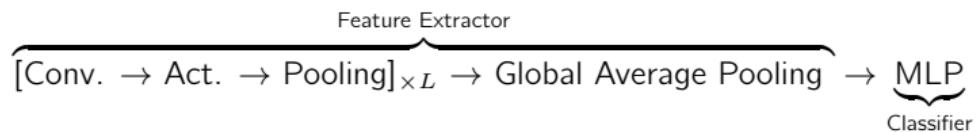
For this example, the top-left output entry of the max pooling remains the same if the input is shifted left and down one step.

How the Size of Receptive Field Evolves



Screenshot from [1]. What happens to the receptive field if we change some of these parameters? Try it at
<https://distill.pub/2019/computing-receptive-fields/>.

Blueprint of CNNs for Classification



Case Study: LeNet-5 [17]

Key Contribution: Pioneer work on using modern CNNs for handwritten character recognition using gradient-based learning.

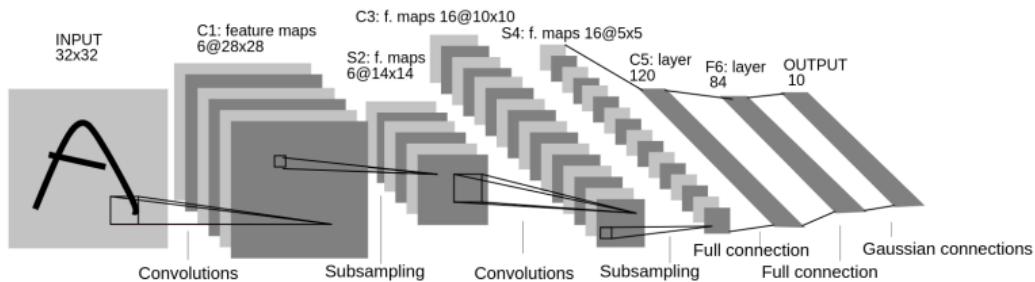


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Building on earlier work on CNNs by LeCun [16] which was inspired by the Neocognitron by Fukushima[5].

Case Study: AlexNet [13]

Key Contributions: First CNN winning ImageNet Challenge (2012); Making use of GPUs in training

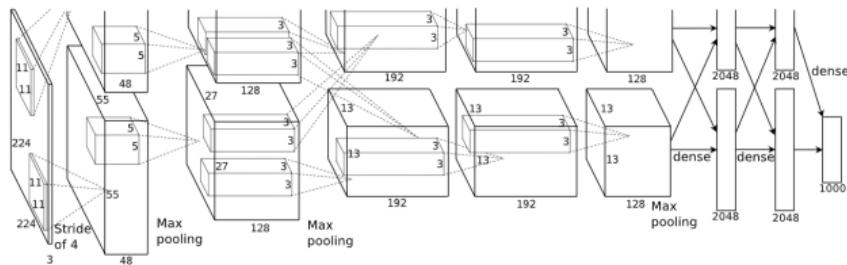
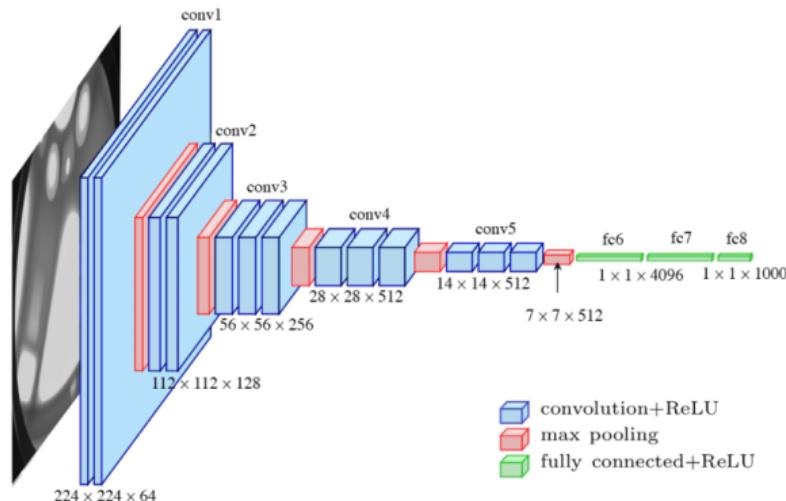


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Case Study: VGG [20]

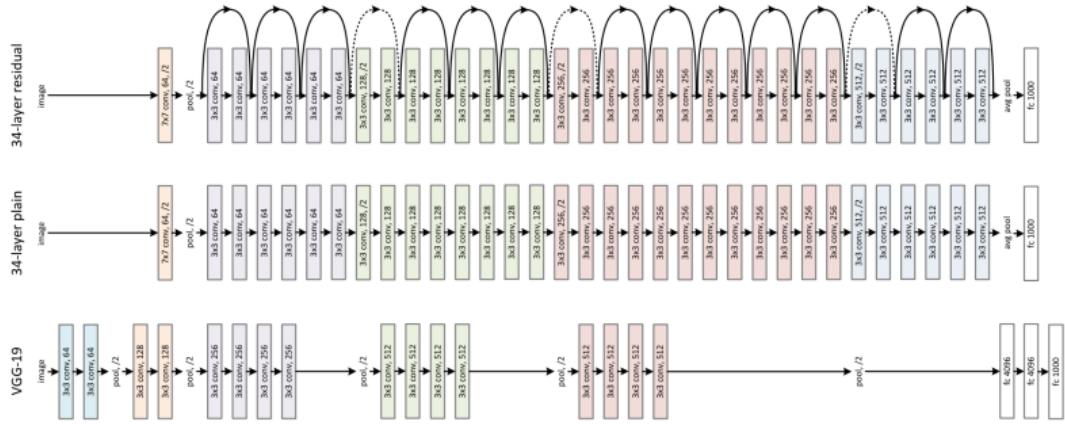
Key Contributions: Winner* of ImageNet Challenge (2014); Demonstrating the benefit of the depth.



: * 1st and 2nd places on the location and classification tracks.

Case Study: ResNets [8]

Key Contributions: Winner of ImageNet Challenge (2015) and many other challenges; Inventing the residue connection that allow training CNNs with many more layers (e.g., 8 \times deeper than VGG).



Case Study: ResNets [8] (cont.)

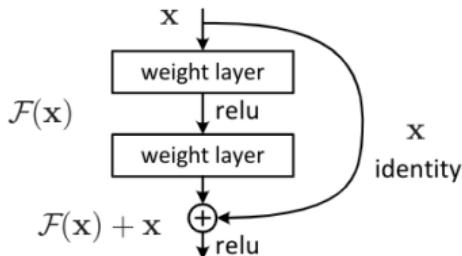


Figure 2. Residual learning: a building block.

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	25.03

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

Residual Connections

- ▶ Core idea of ResNet by He et al.
- ▶ For layer l and intermediate representation x_l with layers NN_l

$$x_{l+1} = x_l + \text{NN}_l(x_l; \theta_l)$$

$$x_{l+2} = x_l + \text{NN}_l(x_l; \theta_l) + \text{NN}_{l+1}(x_{l+1}; \theta_{l+1})$$

...

- ▶ Better gradient flow by 'shortcutting' over high number of intermediate layers between loss and layer NN_l

$$\partial_{\theta_l} x_{l+1} = x_l + \partial_{\theta_l} \text{NN}_l(x_l; \theta_l)$$

$$\partial_{\theta_{l+1}} x_{l+2} = x_l + \text{NN}_l(x_l; \theta_l) + \partial_{\theta_{l+1}} \text{NN}_{l+1}(x_{l+1}; \theta_{l+1})$$

...

- ▶ Allows training far deeper networks with more parameters which result in better performance

What features do CNNs learn? [23]

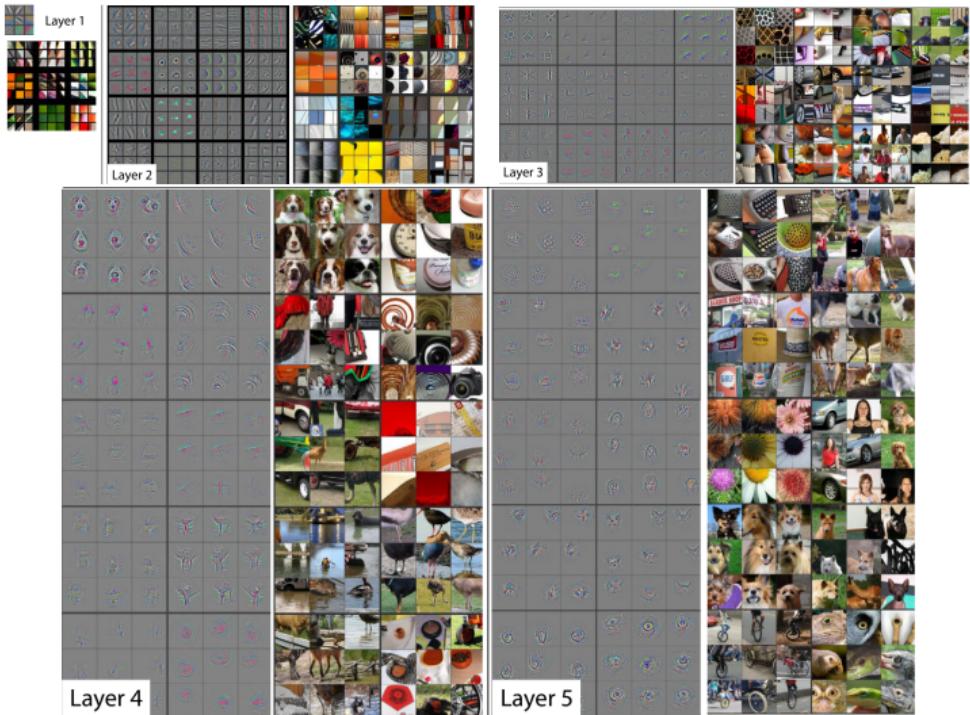
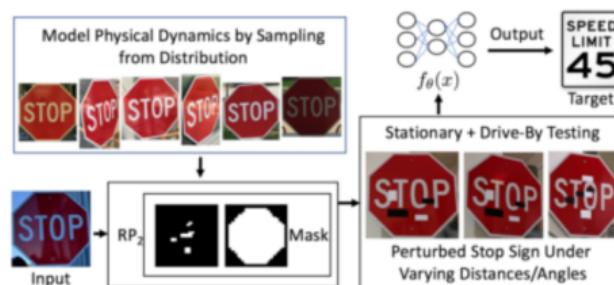


Figure 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are *not* samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form.

Shortcoming of CNNs: Vulnerable to Noise in Input

If the input is slightly (adversarially) perturbed, the prediction of CNNs can dramatically change.

$$\begin{array}{ccc} \text{panda} & + .007 \times & \text{nematode} \\ x & & \text{sign}(\nabla_x J(\theta, x, y)) \\ 57.7\% \text{ confidence} & & 8.2\% \text{ confidence} \\ & = & \\ & & \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \\ & & \text{gibbon} \\ & & 99.3 \% \text{ confidence} \end{array}$$



Top: Adversarial noise causes the prediction of a CNN to change from "Panda" to "Gibbon"; **Bottom:** Physical adversarial sticker causes stop signs to be detected as a target speed limit sign; Figures are from [7, 4] respectively.

Shortcoming of CNNs: Texture Bias [6]

Unlike humans that often rely on shape information in visual processing [14], CNNs rely heavily on textural features.



(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan



(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat



(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

Figure 1: Classification of a standard ResNet-50 of (a) a texture image (elephant skin: only texture cues); (b) a normal image of a cat (with both shape and texture cues), and (c) an image with a texture-shape cue conflict, generated by style transfer between the first two images.

Applications of CNNs

Apart from image classification, CNNs are often used as feature extractors in image-based learning tasks, and the concept of CNNs can be generalized to input from other modalities

- ▶ Different Tasks: Object Detection, Image Segmentation, Image Captioning, ...
- ▶ Other Modalities: Text Classification, Text-to-Speech, ...

Object Detection: You Only Look Once (YOLO) [18]

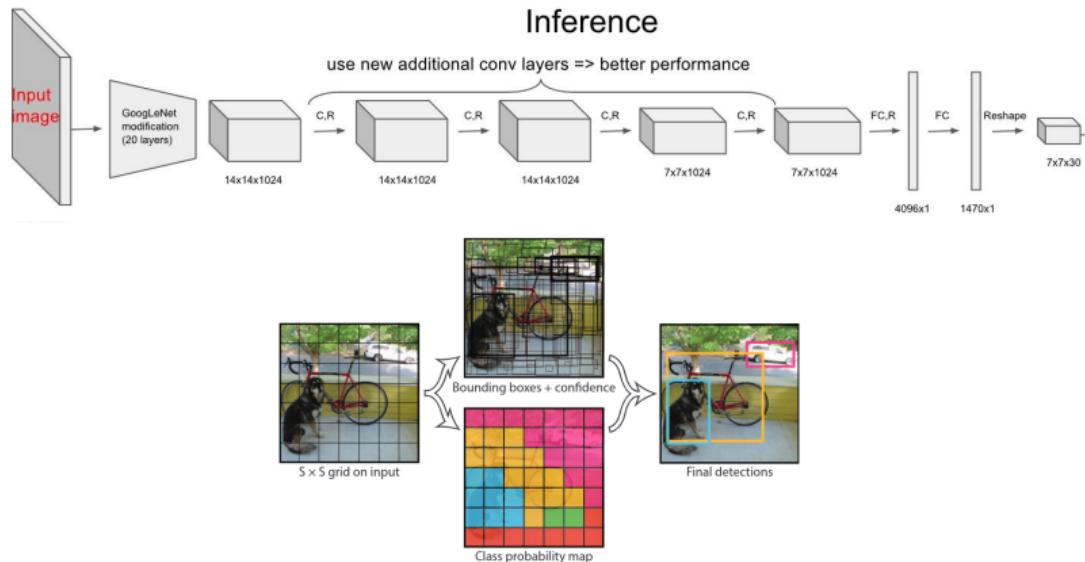
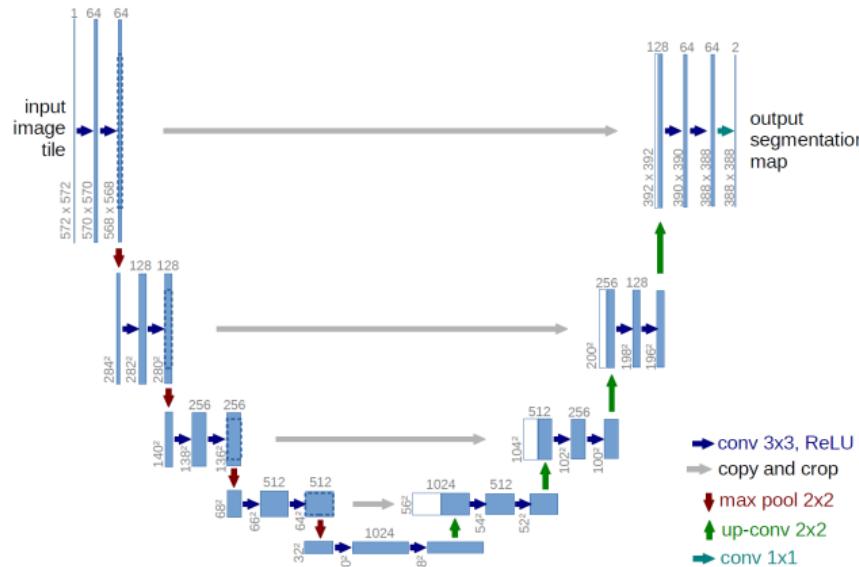


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

The architecture image is from <https://dinghow.site/2019/08/24/object-detection-part1>.

Image Segmentation: U-net [19]

Key Contributions: Fast architecture for precise (biological) image segmentation.



Key concepts: contraction (increase what, reduce where) and expanding paths (precise localization).

Image Segmentation: U-net [19] (cont.)

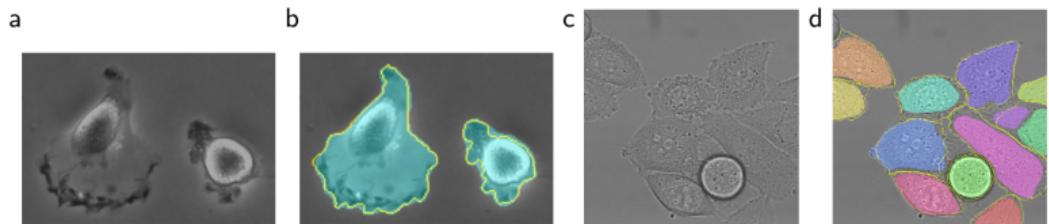
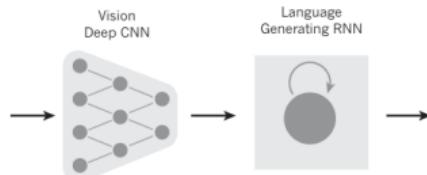


Fig. 4. Result on the ISBI cell tracking challenge. (a) part of an input image of the “PhC-U373” data set. (b) Segmentation result (cyan mask) with manual ground truth (yellow border) (c) input image of the “DIC-HeLa” data set. (d) Segmentation result (random colored masks) with manual ground truth (yellow border).

Image Captioning [21, 10, 22]



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.



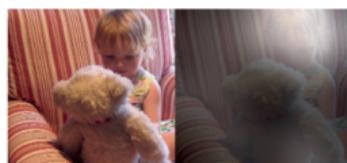
A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.



A group of **people** sitting on a boat in the water.



A **giraffe** standing in a forest with **trees** in the background.

Figure from [22].

Convolution and Pooling for Text Classification [11]

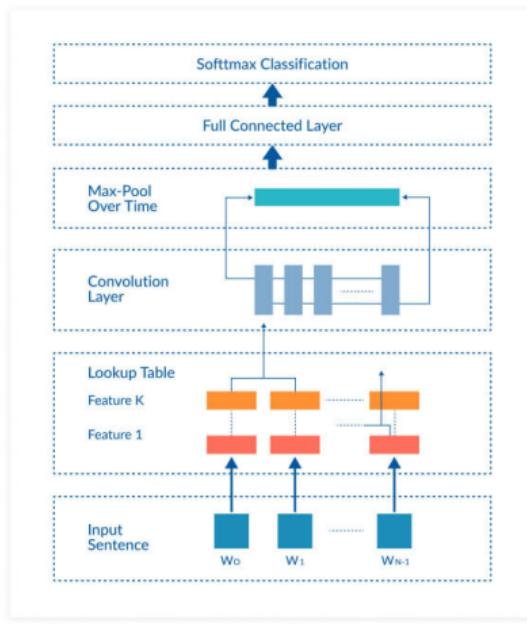
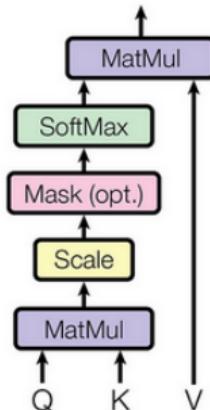


Figure is from [https://indiantechwarrior.com/
sentence-classification-using-convolutional-neural-networks/](https://indiantechwarrior.com/sentence-classification-using-convolutional-neural-networks/)

Transformers: Attention Mechanism

- ▶ Idea: find global interactions within an input sequence or between two input sequences.
- ▶ In our case: $K = V = x \in \mathbb{R}^T$ and $Q = y \in \mathbb{R}^{T'}$
- ▶ "Query with target $y_{t'}$ all source keys x_t on how much attention to pay to source value x_t at each target timestep t' "
- ▶ Compute pairwise product $QK^T \in \mathbb{R}^{T' \times T}$ to obtain similarities
- ▶ Normalize over source dimension with softmax probability
- ▶ Scale source values $V = x$

$$\underbrace{\text{Attention}(Q, K, V)}_{T' \times T} = \underbrace{\text{Softmax}_T(QK^T)}_{T' \times T} V \quad (1)$$



Transformers: Attention Mechanism

- ▶ Original attention mechanism in Bahdanau et al used MLP as scalar projector
- ▶ QK^T is a dot product, so we can use arbitrary dimensions
- ▶ $x \in \mathbb{R}^{T \times F}$ and $y \in \mathbb{R}^{T' \times F}$, $QK^T \in \mathbb{R}^{T \times T'}$

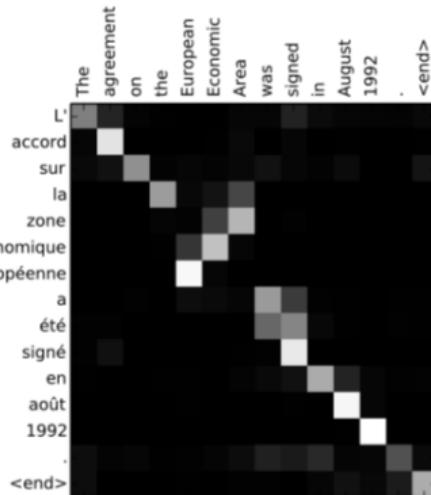


Figure: "Neural Machine Translation by Jointly Learning to Align and Translate" by Bahdanau et al.

Summary

- ▶ CNNs exploit local structure of (2d) and learn hierarchical representation for a given task.
- ▶ CNNs share parameters between spatial locations, and they are thus suitable to learn from signal where features potentially appear in any location.
- ▶ Main ingredients of CNNs: Convolution and Pooling layers.
- ▶ CNNs are widely used in many applications and domains (beyond image data).
- ▶ CNNs force local interactions in the first layers. For some tasks, it is beneficial to capture global interactions (this can be achieved with attention layers).

Bibliography I

- [1] A. Araujo, W. Norris, and J. Sim.
Computing receptive fields of convolutional neural networks.
Distill, 2019.
<https://distill.pub/2019/computing-receptive-fields>.
- [2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.
ImageNet: A Large-Scale Hierarchical Image Database.
In *CVPR09*, 2009.
- [3] V. Dumoulin and F. Visin.
A guide to convolution arithmetic for deep learning.
arXiv preprint arXiv:1603.07285, 2016.
- [4] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song.
Robust physical-world attacks on deep learning visual classification.
In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1625–1634. Computer Vision Foundation / IEEE Computer Society, 2018.
- [5] K. Fukushima.
A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.
Biol. Cybern., 36:193–202, 1980.
- [6] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel.
Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness.
In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [7] I. J. Goodfellow, J. Shlens, and C. Szegedy.
Explaining and harnessing adversarial examples.
In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Bibliography II

- [8] K. He, X. Zhang, S. Ren, and J. Sun.
Deep residual learning for image recognition.
In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.
- [9] S. Hooker.
The hardware lottery.
Communications of the ACM, 64(12):58–65, 2021.
- [10] A. Karpathy and L. Fei-Fei.
Deep visual-semantic alignments for generating image descriptions.
In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3128–3137, 2015.
- [11] Y. Kim.
Convolutional neural networks for sentence classification.
In A. Moschitti, B. Pang, and W. Daelemans, editors, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1746–1751. ACL, 2014.
- [12] J. Krause, M. Stark, J. Deng, and L. Fei-Fei.
3d object representations for fine-grained categorization.
In 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia, 2013.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton.
Imagenet classification with deep convolutional neural networks.
In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States, pages 1106–1114, 2012.
- [14] B. Landau, L. B. Smith, and S. S. Jones.
The importance of shape in early lexical learning.
Cognitive development, 3(3):299–321, 1988.

Bibliography III

- [15] Y. LeCun, Y. Bengio, and G. Hinton.
Deep learning.
nature, 521(7553):436–444, 2015.
- [16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel.
Backpropagation applied to handwritten zip code recognition.
Neural computation, 1(4):541–551, 1989.
- [17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.
- [18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi.
You only look once: Unified, real-time object detection.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [19] O. Ronneberger, P. Fischer, and T. Brox.
U-net: Convolutional networks for biomedical image segmentation.
In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [20] K. Simonyan and A. Zisserman.
Very deep convolutional networks for large-scale image recognition.
In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- [21] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan.
Show and tell: A neural image caption generator.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.
- [22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio.
Show, attend and tell: Neural image caption generation with visual attention.
In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

Bibliography IV

- [23] M. D. Zeiler and R. Fergus.
Visualizing and understanding convolutional networks.
In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014.