Università degli Studi di Padova

Machine Learning 2024/2025

AMCO
Artificial Intelligence, Machine Learning and Control Research Group

Lecture #27 Convolutional Neural Networks & Autoencoders

Gian Antonio Susto

Before starting: last lectures

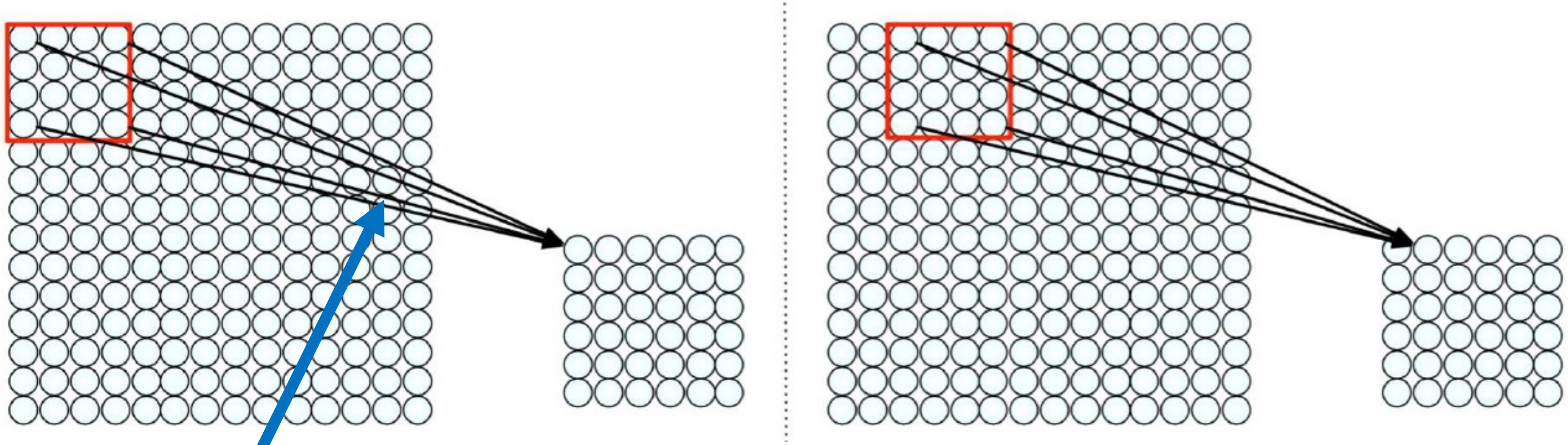
WEEK 12
2025-05-12 Monday – Lecture 29: Fairness in ML
2025-05-15 Thursday – (Optional) Programming Mock Exam discussion
2025-05-16 Friday - Lecture 30 (Lab 09): NN #02

WEEK 13
2025-05-19 Monday – (Optional) Theory recap session with TAs
2025-05-22 Thursday – (No exam) Lecture 31: Real-world Applications and MLOps
2025-05-23 Friday – Lecture 32 (Lab 10): Recap LAB + Exam sim

WEEK 14
2025-05-26 Monday – Lecture 33: XAI #01
2025-05-29 Thursday – Lecture 34: XAI #02
2025-05-30 Friday – Lecture 35 (Lab 11): XAI LAB

WEEK 15
2025-06-05 Thursday – (No exam) Lecture 36: ML, what's next?

Recap: 2D convolutions



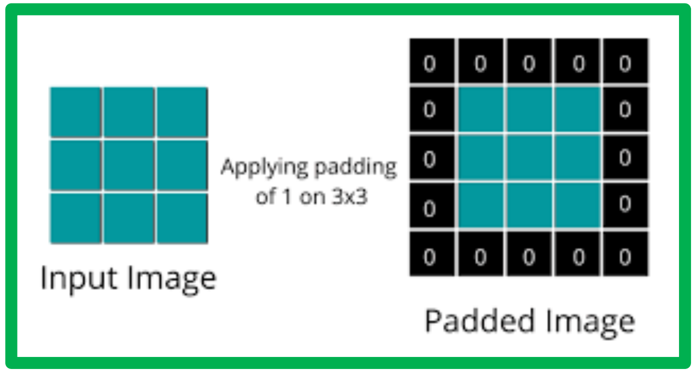
1	2	3
4	5	6
7	8	9

1	2	1
0	0	0
-1	-2	-1

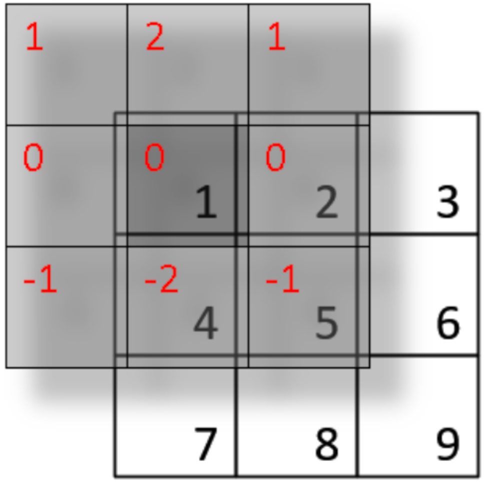
The operation is the 2D convolution: we multiply element-wise 2 matrices (the input and the kernel – weights), then we sum and derive some 'features'

From https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionO

From http://cs231n.github.io/convolutional-networks/



[Optional procedure] We include 'padding' the image (equal to 1): we add a frame around the image of pixels with value = 0. When performing the convolution operation, padding allow border pixels to have similar importance to inner pixels



$$
\begin{aligned}
y[0,0] &= \sum_j \sum_i x[i,j] \cdot h[0-i, 0-j] \\
&= \quad x[-1,-1] \cdot h[1,1] + x[0,-1] \cdot h[0,1] + x[1,-1] \cdot h[-1,1] \\
&\quad + x[-1,0] \cdot h[1,0] \quad + x[0,0] \cdot h[0,0] \quad + x[1,0] \cdot h[-1,0] \\
&\quad + x[-1,1] \cdot h[1,-1] + x[0,1] \cdot h[0,-1] + x[1,1] \cdot h[-1,-1] \\
&= \quad 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 \\
&\quad + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 \\
&\quad + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1) \\
&= -13
\end{aligned}
$$

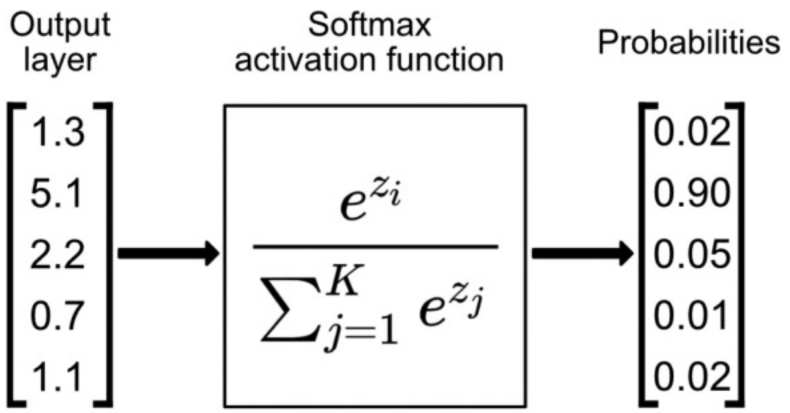
From https://www.songho.ca/dsp/convolution/convolution2d_example.html

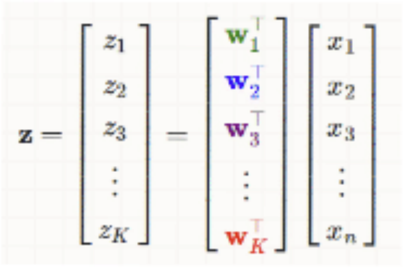
Soft-max layer

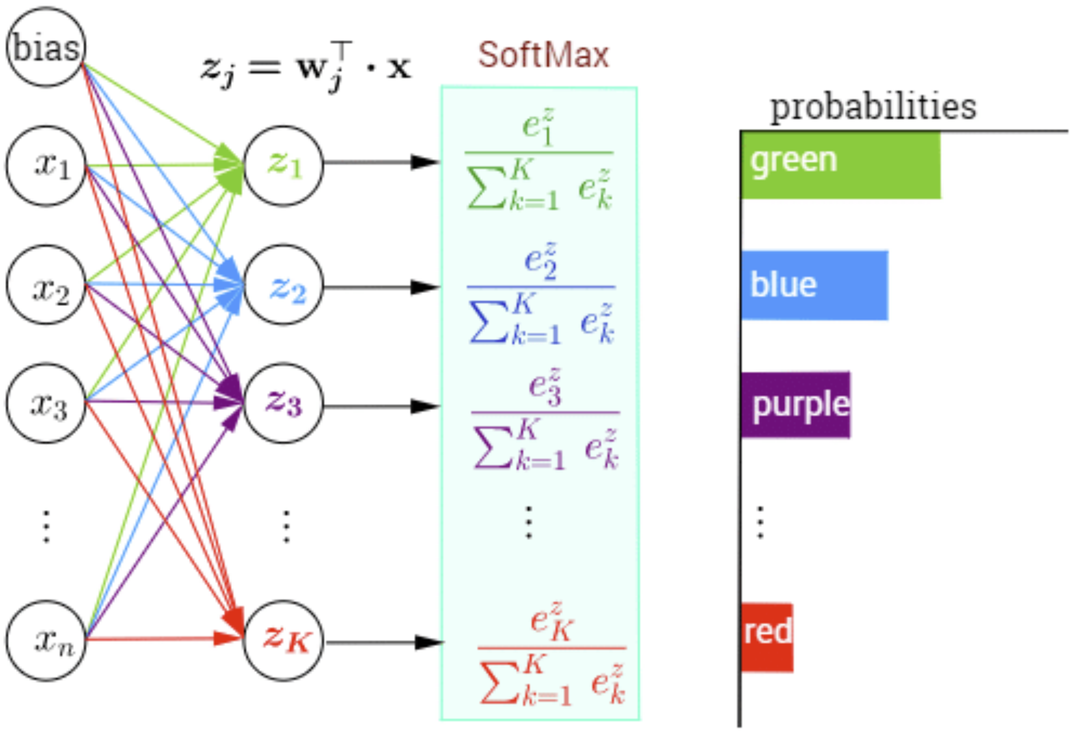
- Softmax turns raw scores (logits) into probabilities.
- Each output is in the range (0, 1), and all outputs sum to 1.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

- Commonly used in the final layer of neural networks for classification.







Cross-entropy loss

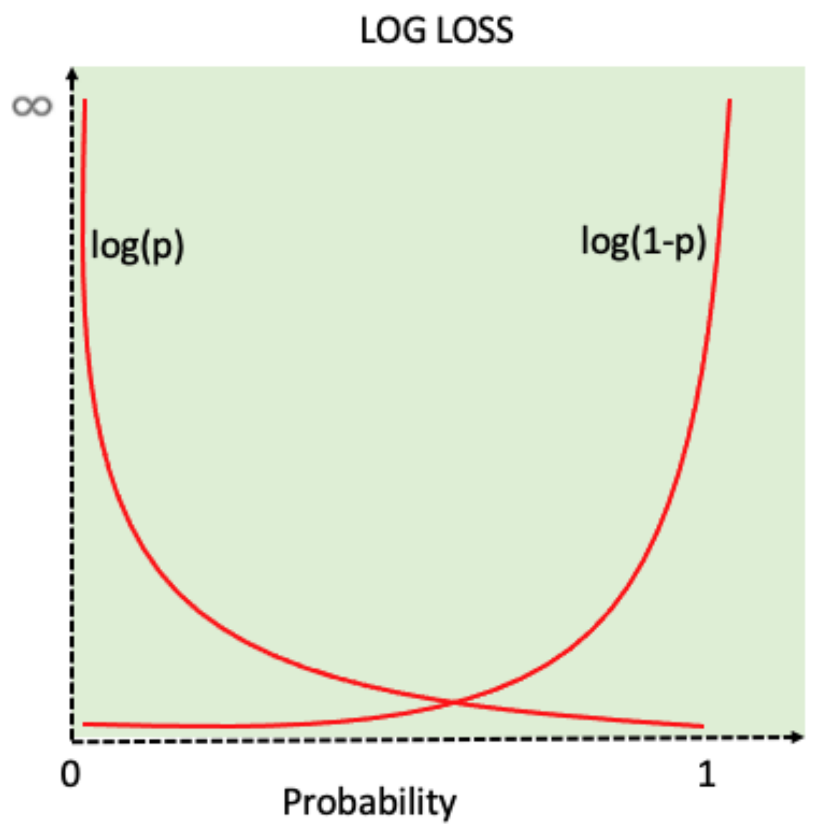
- Generalization of log-likelihood seen in logistic regression
- Measures the difference between predicted probabilities and true labels.
- Used as a loss function in classification tasks.
- Formula (binary):

$$-[y \cdot \log(p) + (1 - y) \cdot \log(1 - p)]$$

- Formula (multi-class):

$$-\sum_i y_i \cdot \log(p_i)$$

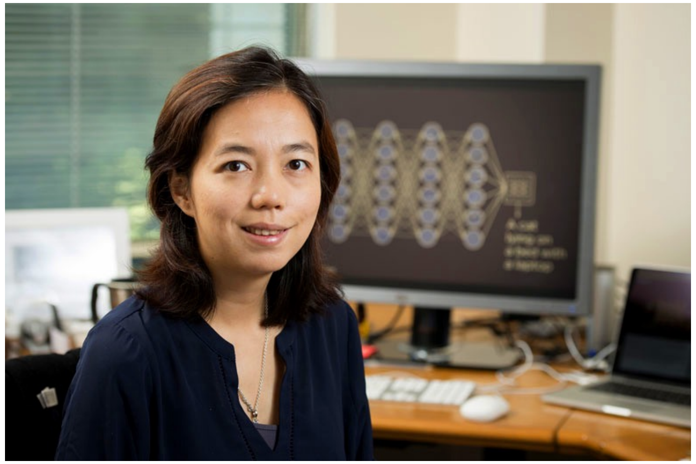
- Lower values mean better predictions.



Imagenet (2009)

- 21841 classes
- 14M images with different dimensions and resolutions (many apply resize to 256x256)
- Unbalanced dataset
- Colour images
- Lead developer Fei-Fei Li

IMAGENET



"Elongated crescent-shaped yellow fruit with soft sweet flesh"

1409 pictures of bananas.

MIT https://groups.csail.mit.edu/vision/TinyImages/

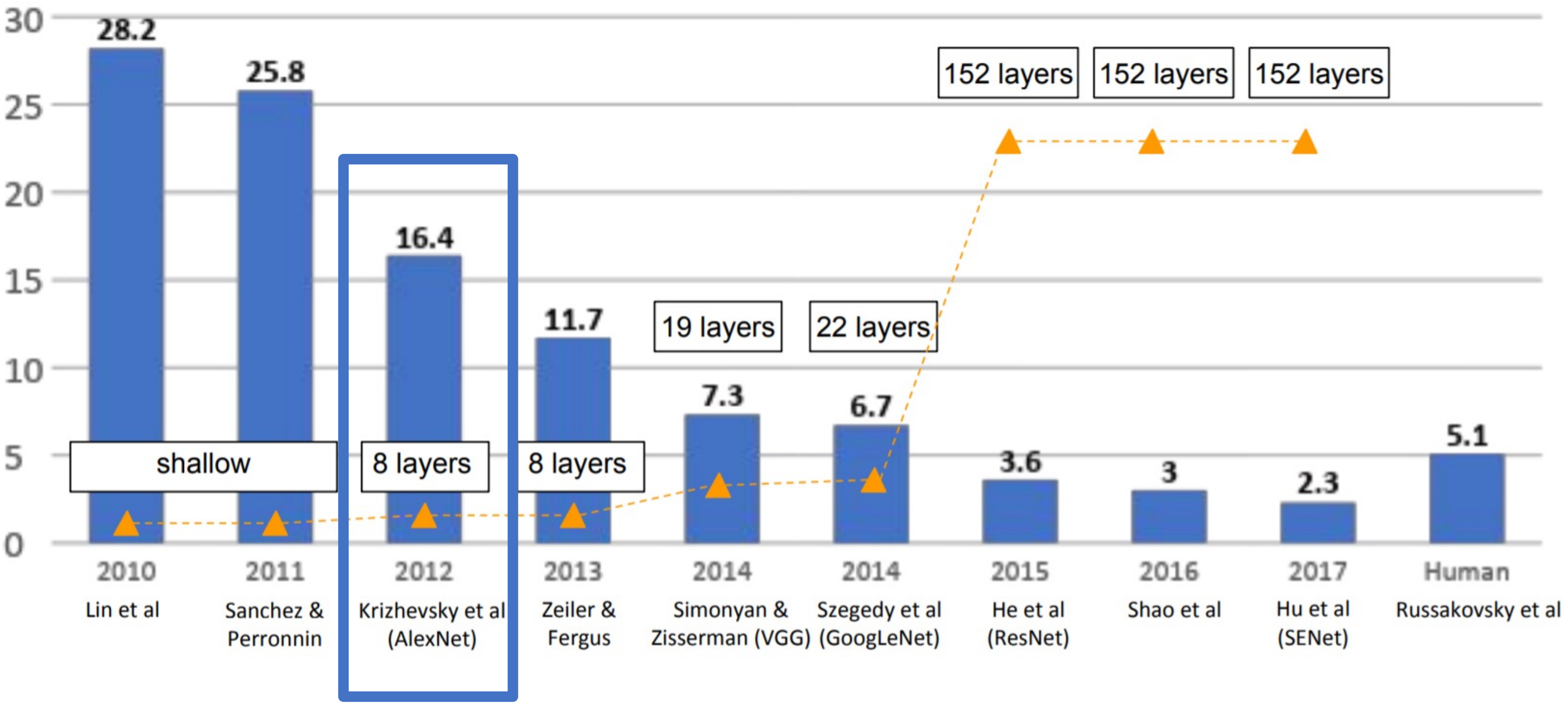
Classification task: produce a list of object categories present in image. 1000 categories.

“Top 5 error”: rate at which the model does not output correct label in top 5 predictions

An overview on the most famous architectures

Imagenet – visual recognition challenge with 1000 classes.

Winners:



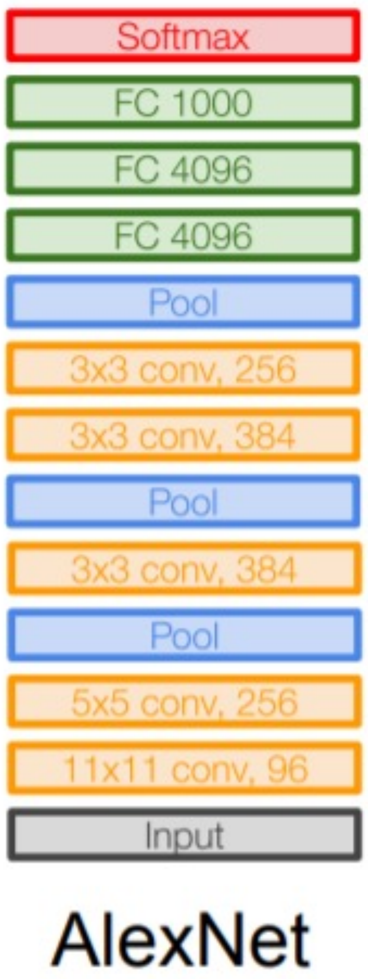
First CNN-based winner

AlexNet

AlexNet (2012) renewed interest in CNN.

It uses:

- RELU (first use)
- Layer normalization
- Dropout
- MaxPooling
- Momentum
- Data augmentation in training



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

From: http://cs231n.stanford.edu - Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

AlexNet

AlexNet (2012) renewed interest in CNN.

It uses:

- RELU (first use)
- Layer normalization
- Dropout
- MaxPooling
- Momentum
- **Data augmentation in training**

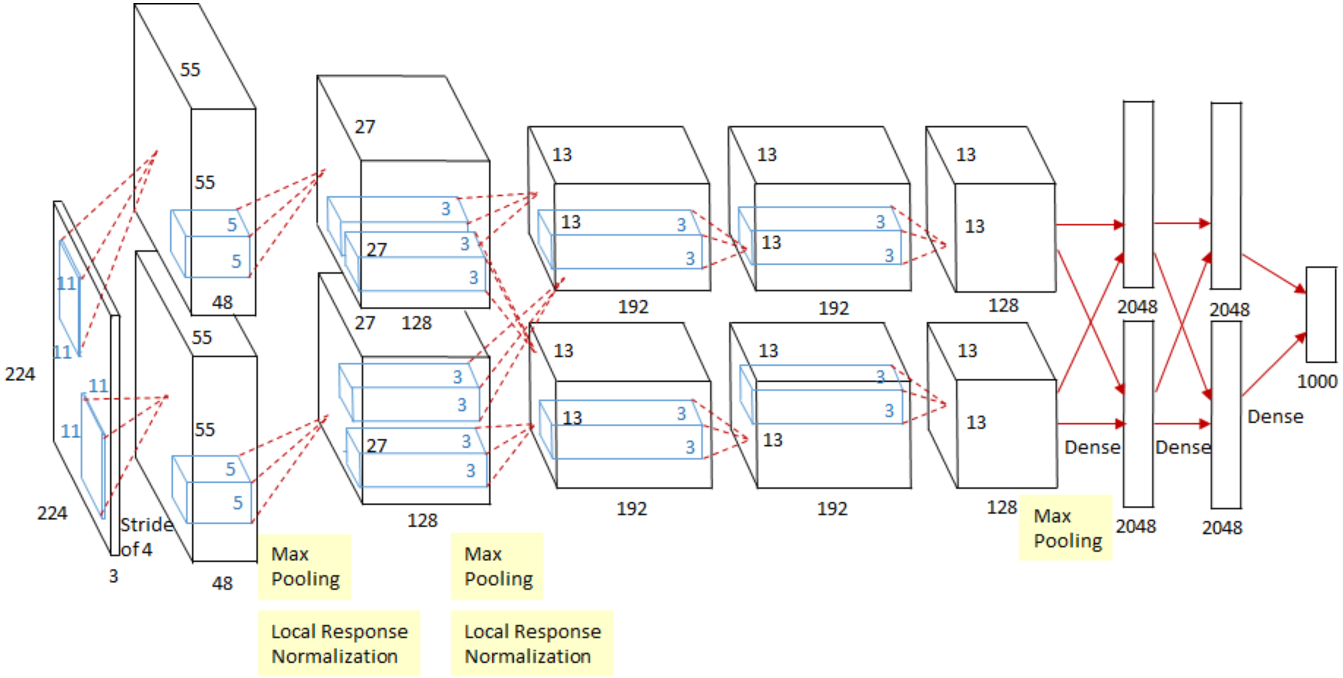
From: http://cs231n.stanford.edu - Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

AlexNet

AlexNet (2012) renewed interest in CNN.

It uses:

- RELU (first use)
- Layer normalization
- Dropout
- MaxPooling
- Momentum
- **Data augmentation in training**



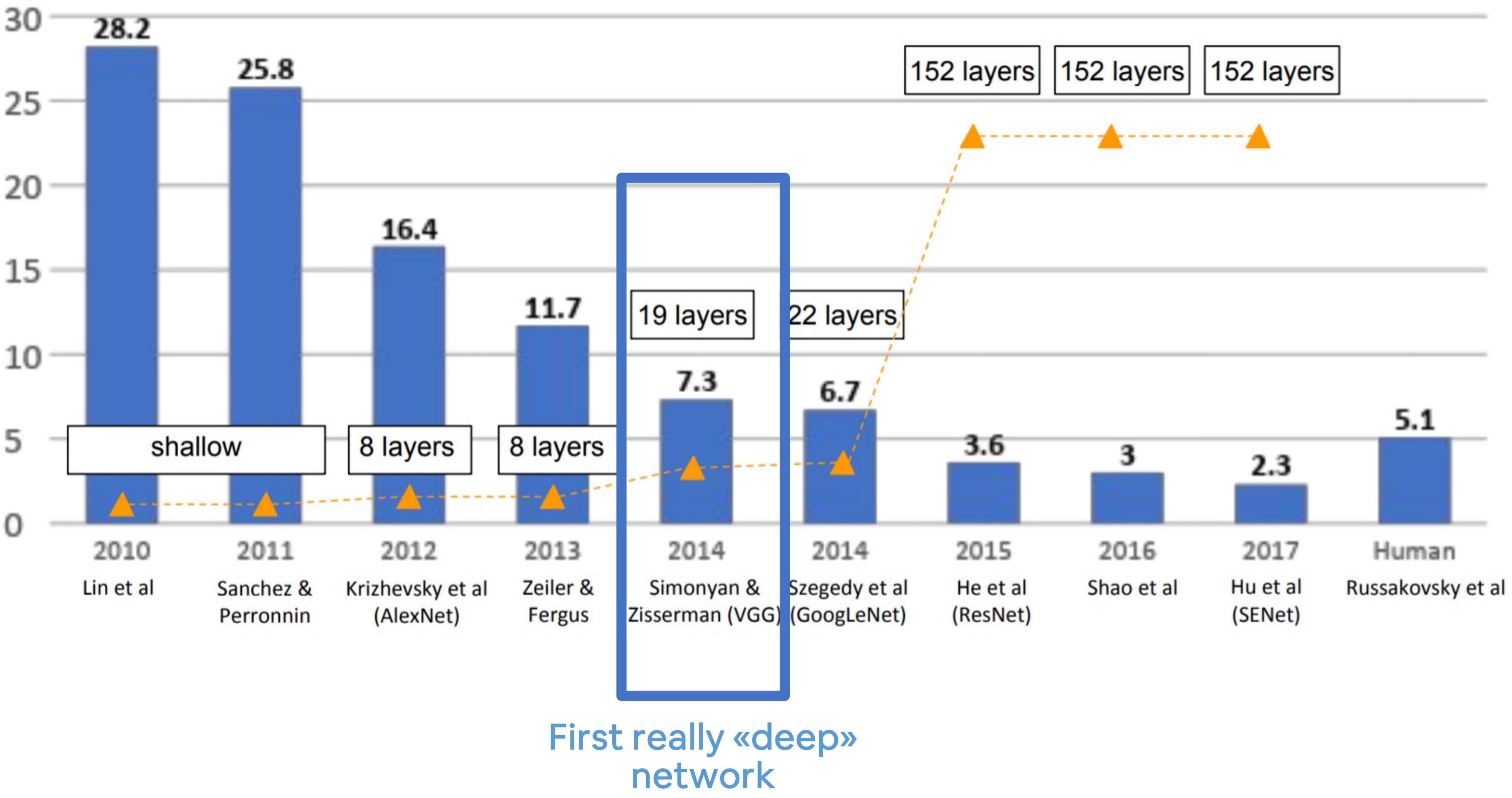
62.3 million parameters: the original paper showed that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of (GPUs) during training.

From: http://cs231n.stanford.edu - Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

An overview on the most famous architectures

Imagenet – visual recognition challenge with 1000 classes.

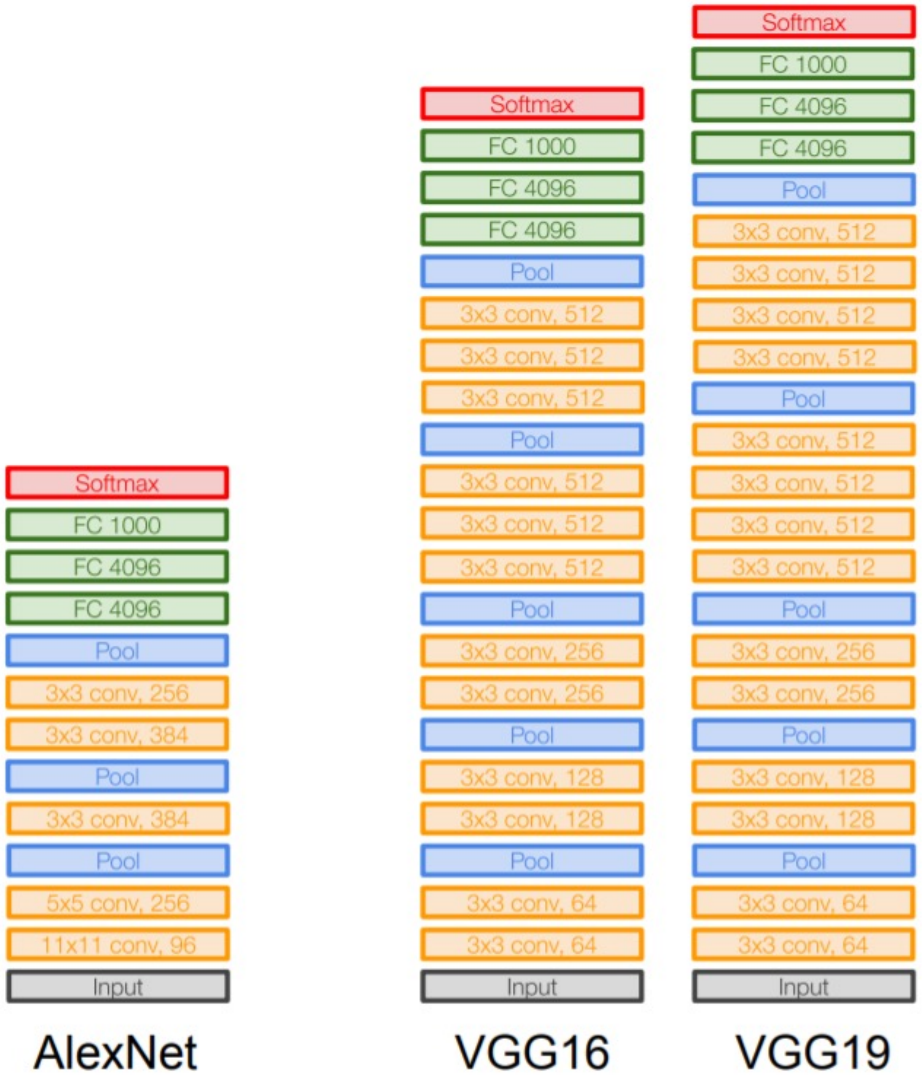
Winners:



VGG

VGG (2014):

- many small convolutional filters stacked together before pooling
- very deep (at the time) with 16/19 layers



From: http://cs231n.stanford.edu

VGG

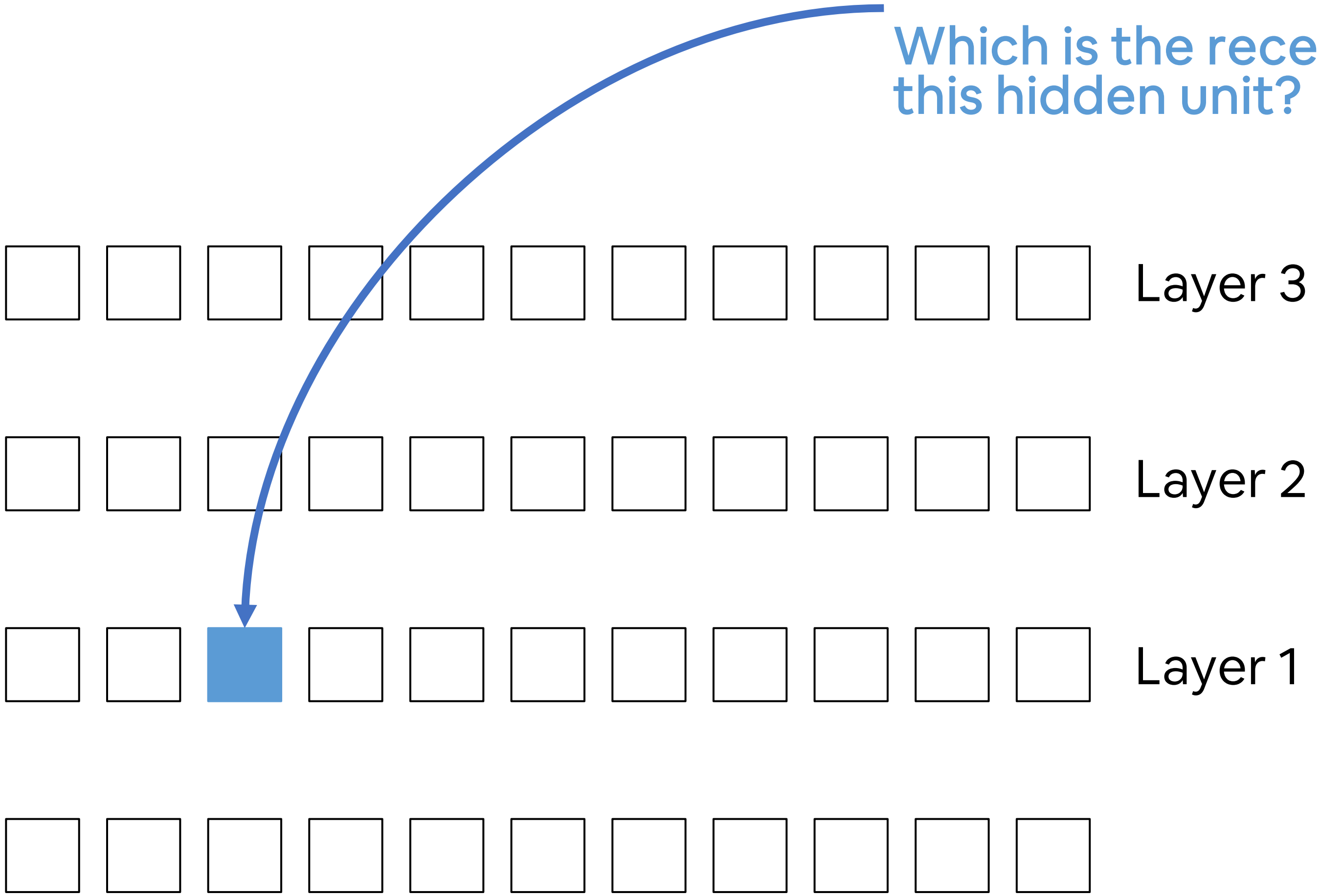
Wait. Why using smaller conv filters?

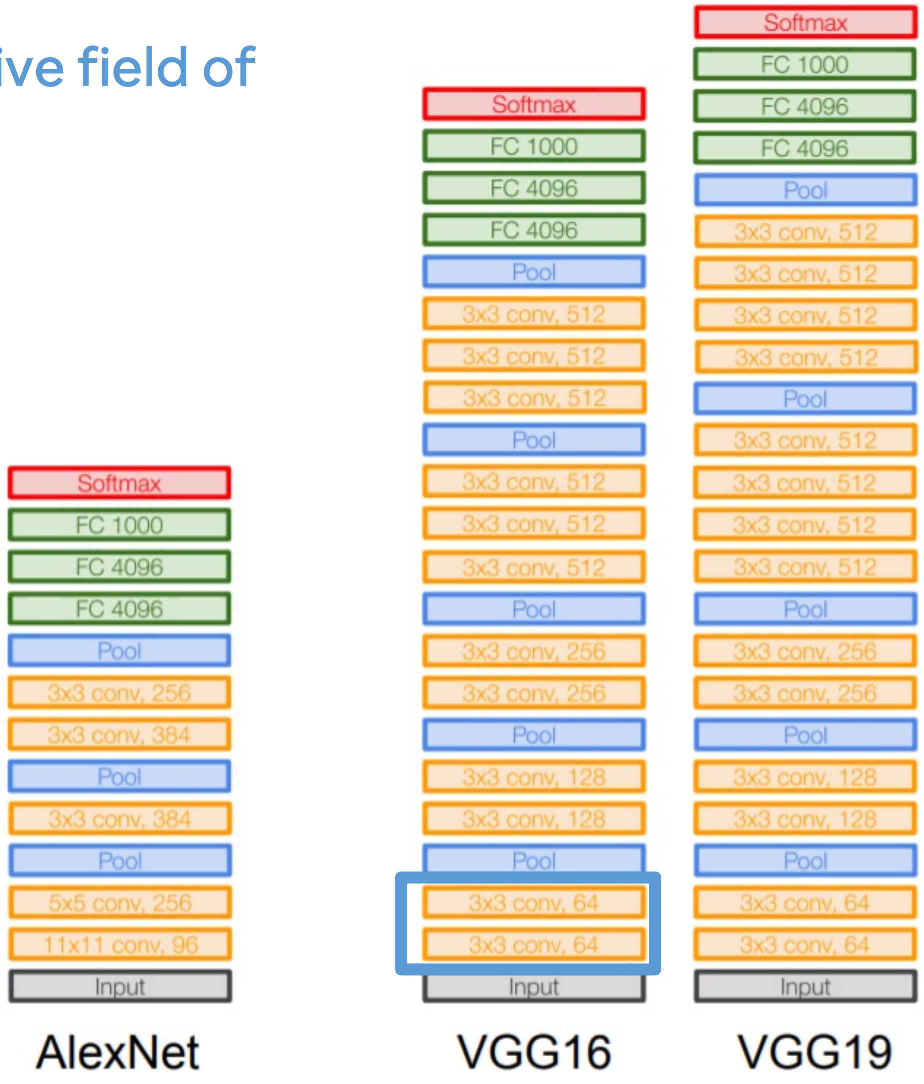
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?



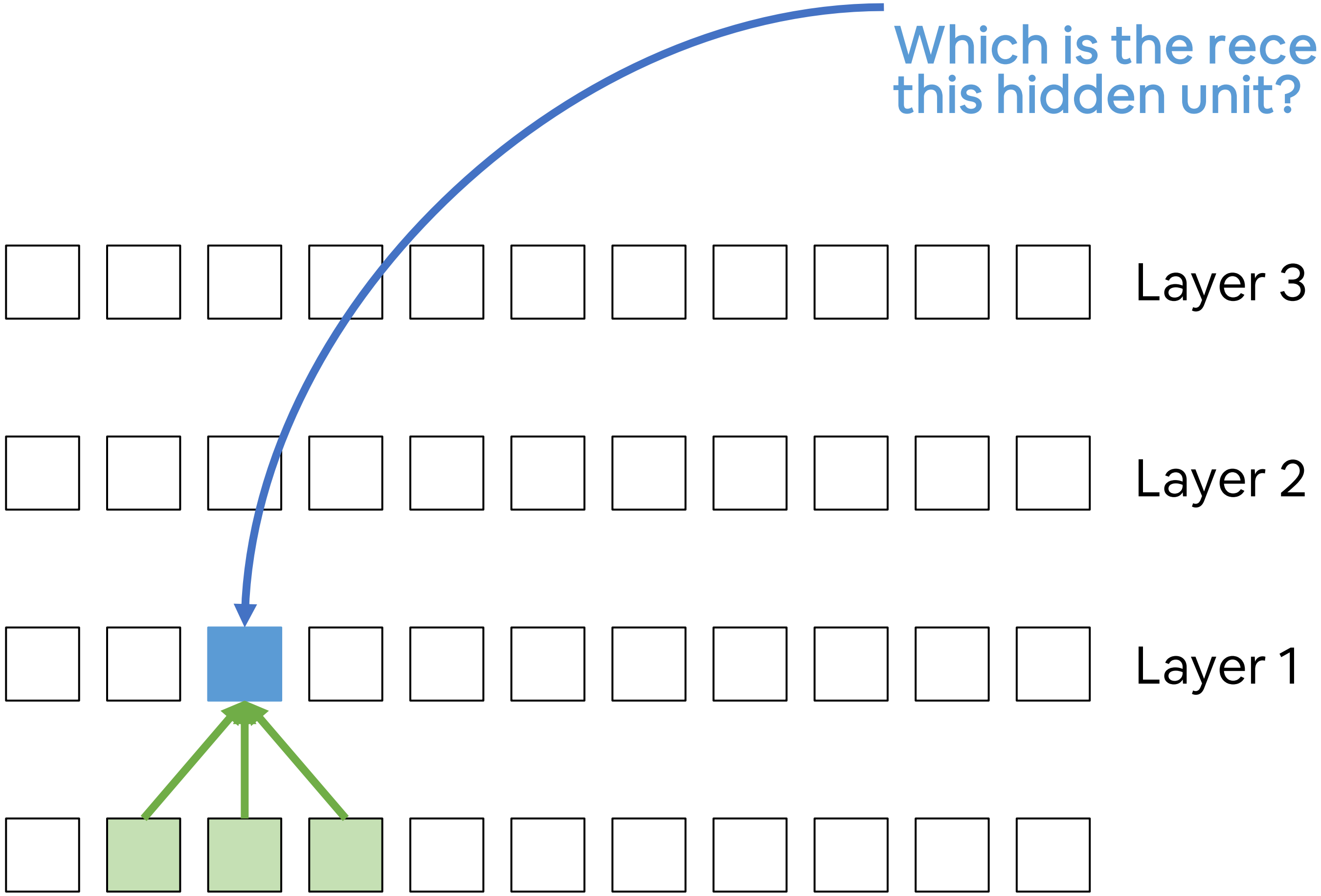


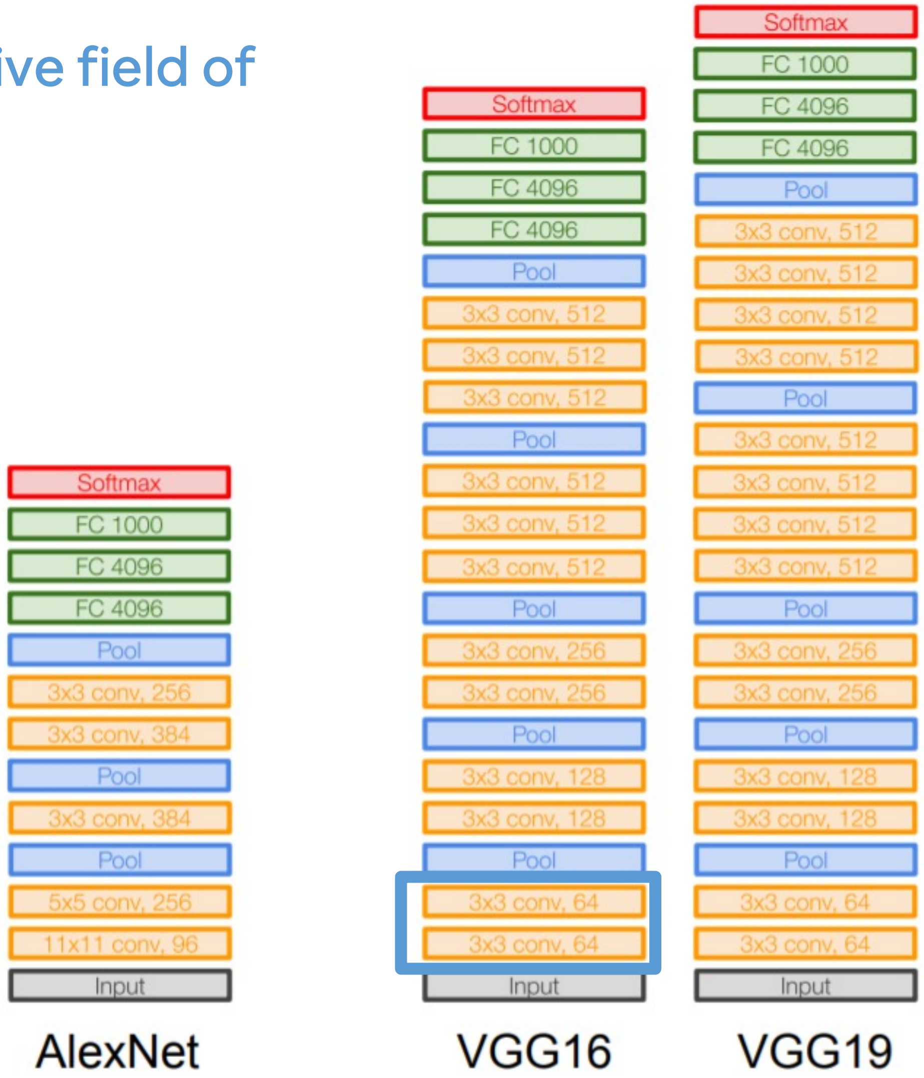
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?



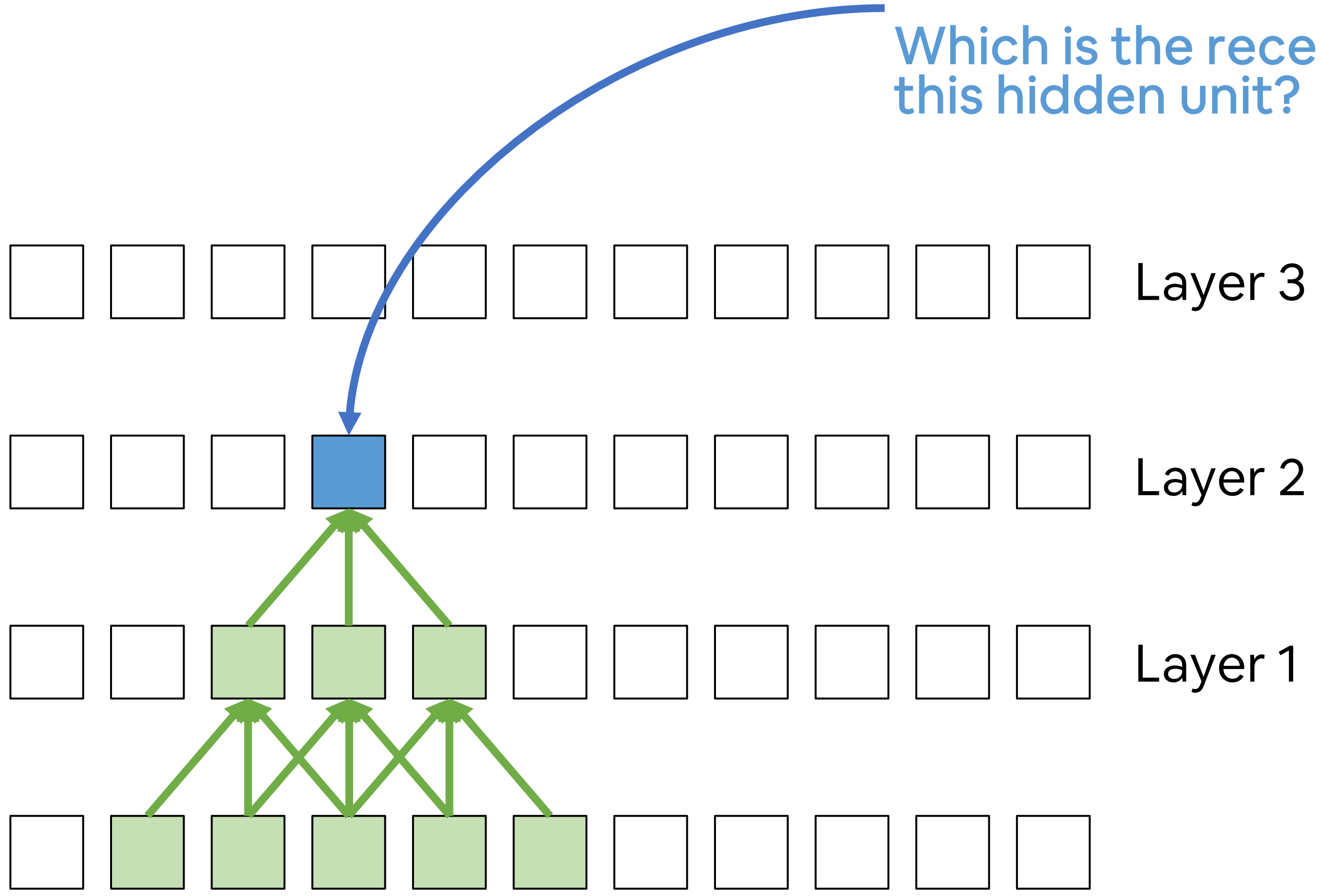


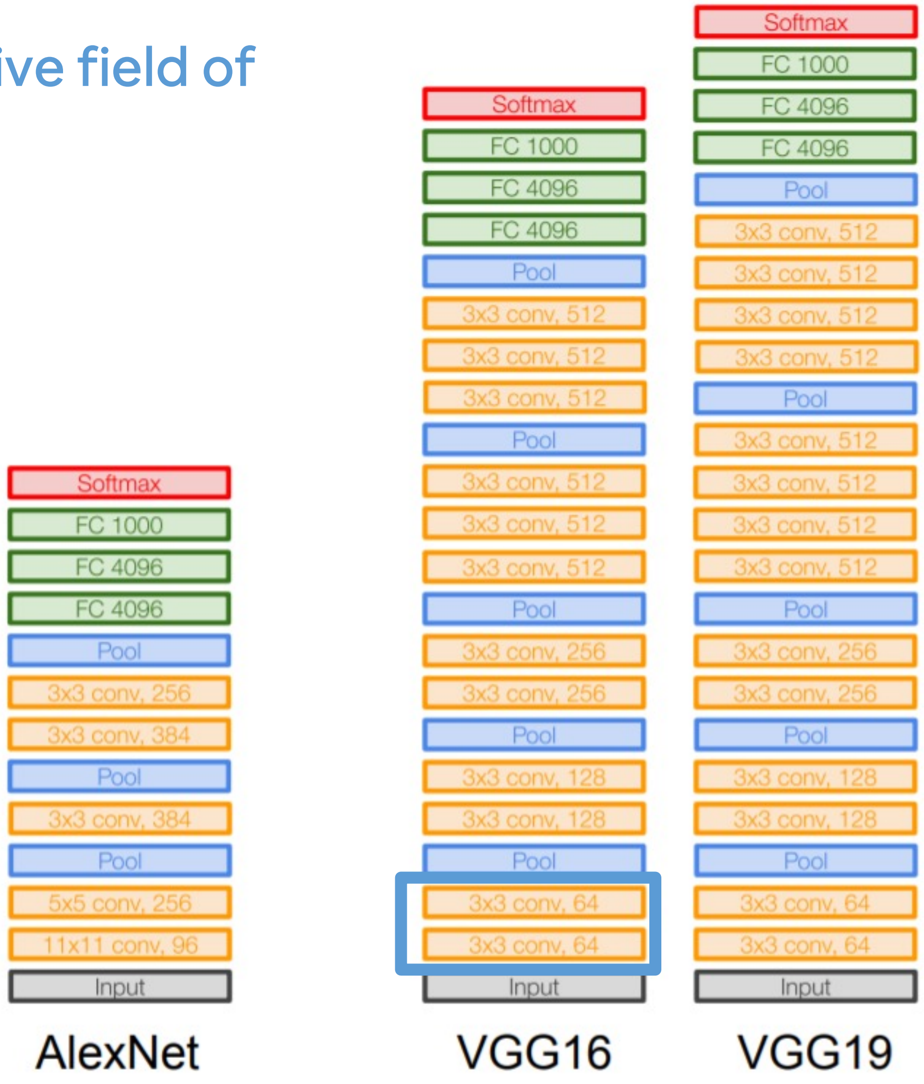
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?



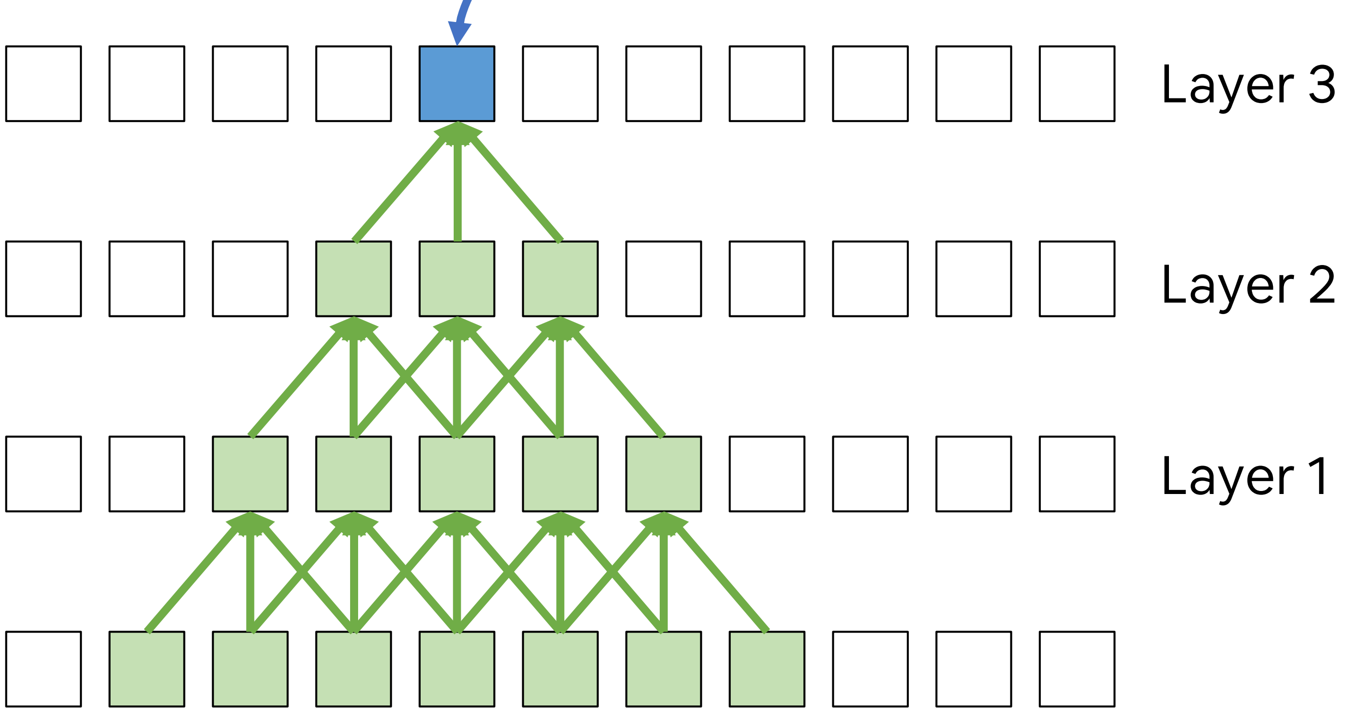


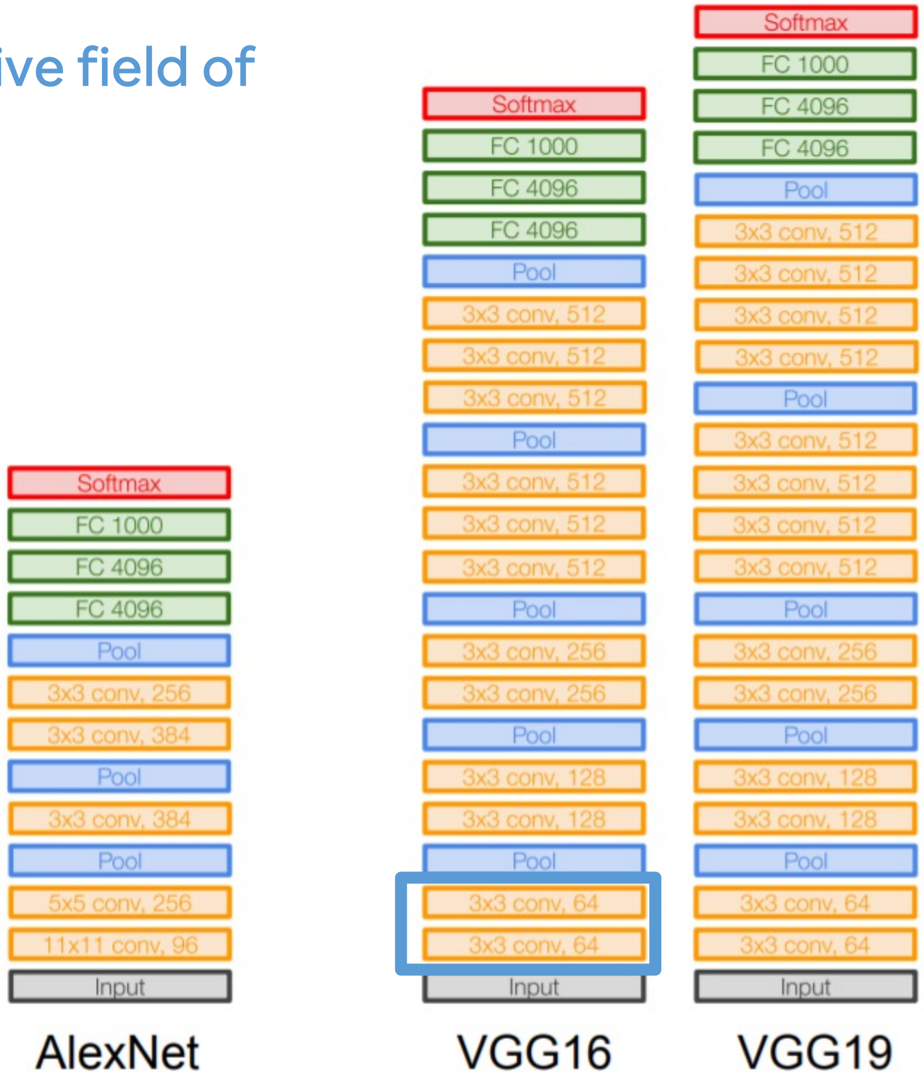
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?





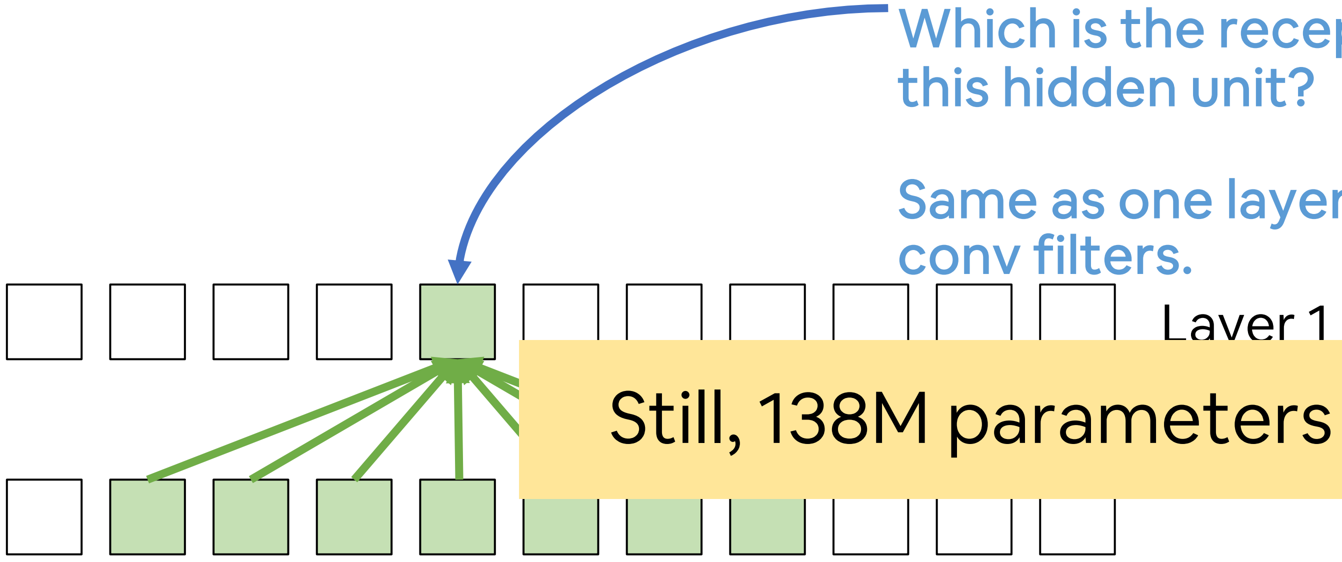
From: http://cs231n.stanford.edu

VGG

Consider a stack of three 3x3 conv layers.

Which is the receptive field of this hidden unit?

Same as one layer with 7x7 conv filters.



Still, 138M parameters for VGG16!

From: http://cs231n.stanford.edu

But three layers mean more non-linearities, i.e. more complex features...

... and fewer parameters!

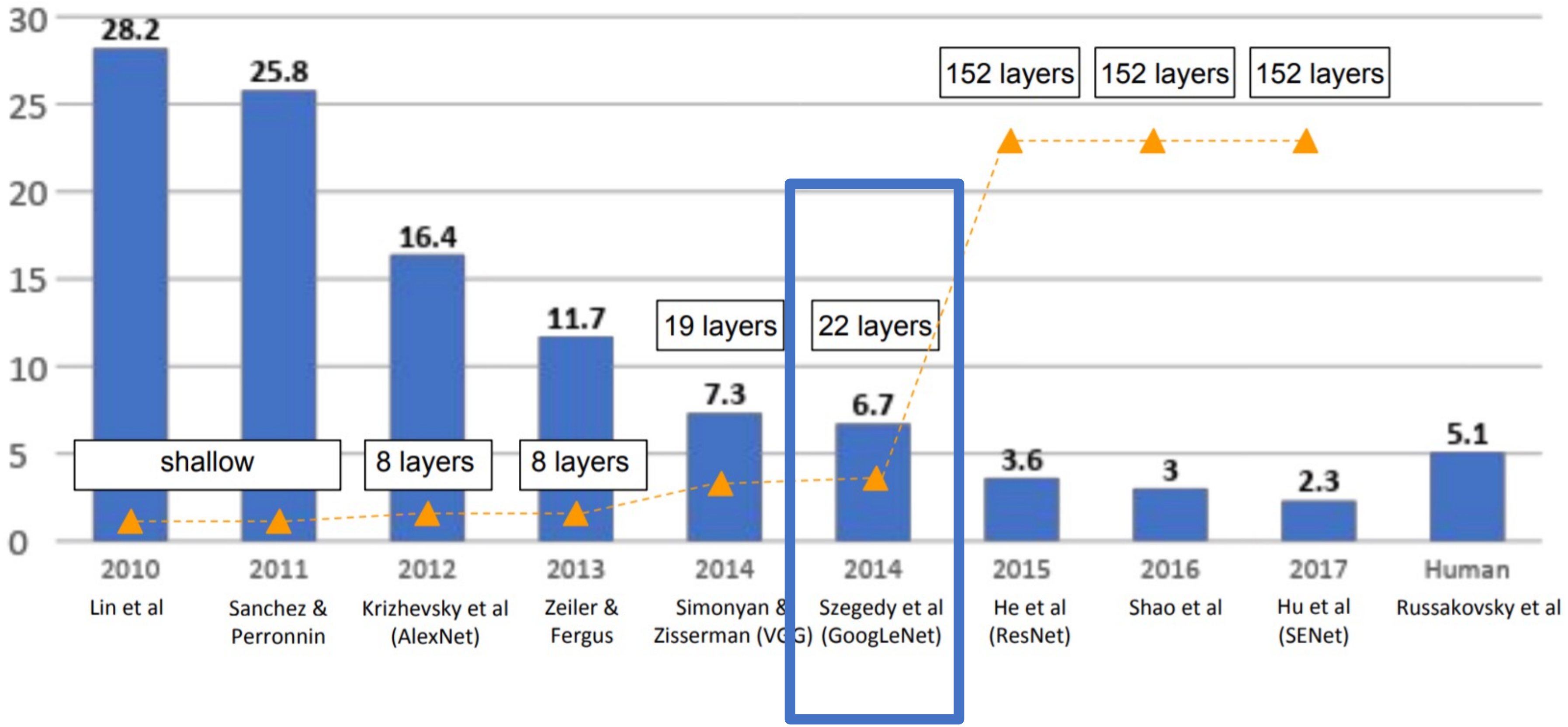
C channels (e.g. C=3 for RGB images):

- 7x7 conv has $7^2 \times C = 147$ parameters
- 3 layers of 3x3 conv have $3 \times (3^2 \times C) = 81$

An overview on the most famous architectures

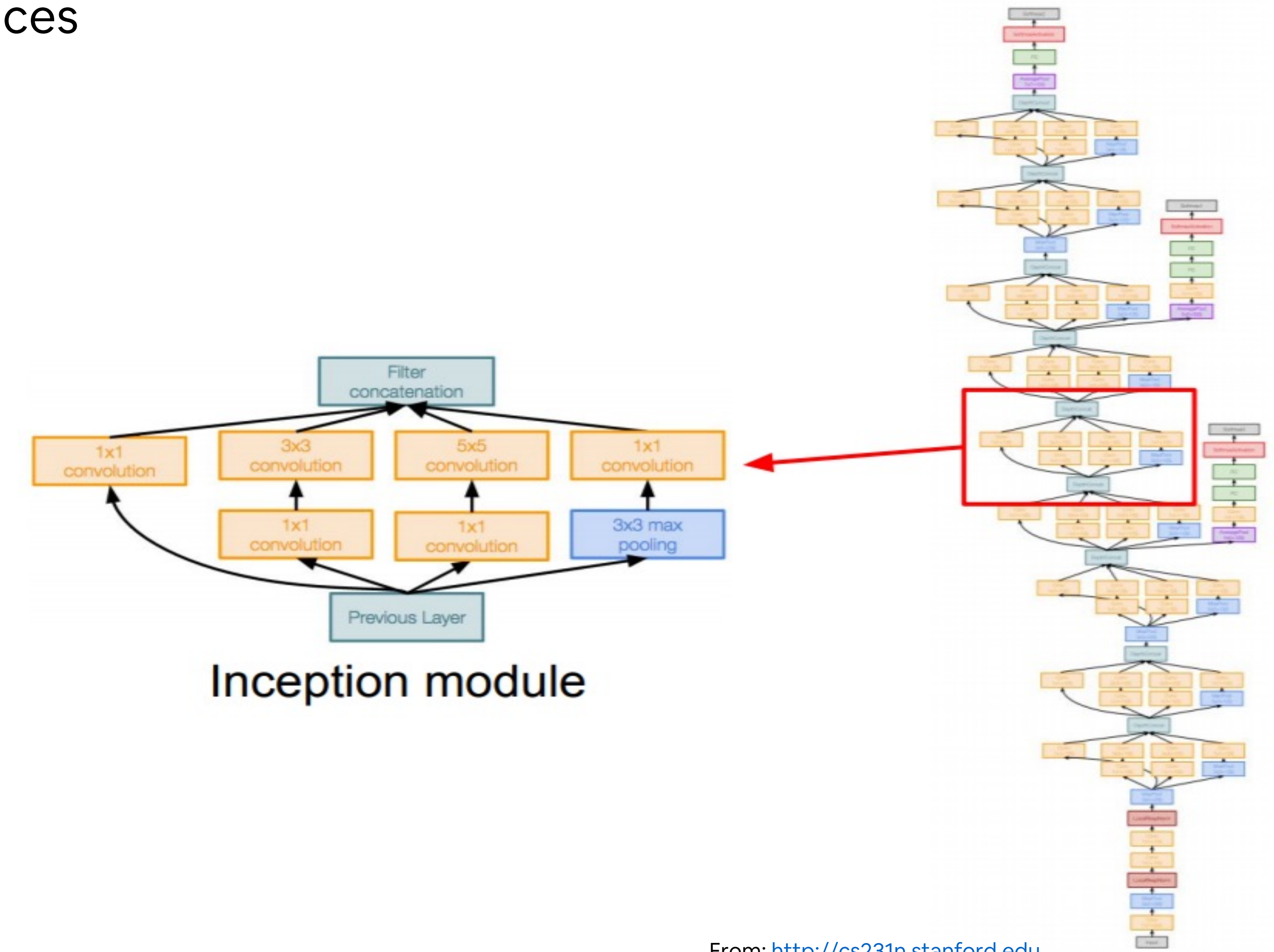
Imagenet – visual recognition challenge with 1000 classes.

Winners:



GoogLeNet (optional)

Inception + GoogLeNet (2015)– introduces parallel conv blocks (inception)



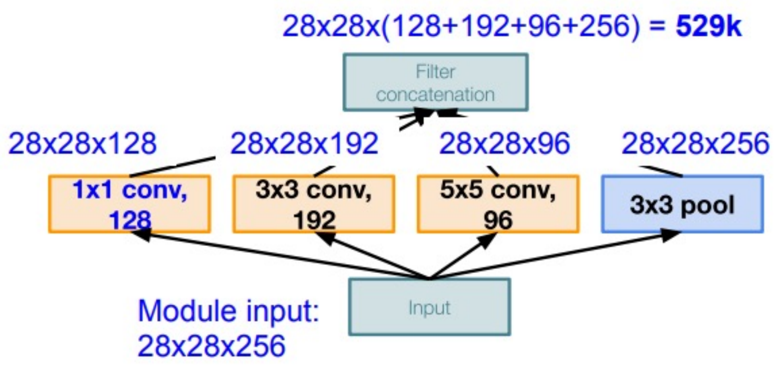
From: http://cs231n.stanford.edu

From: http://cs231n.stanford.edu

GoogLeNet (optional)

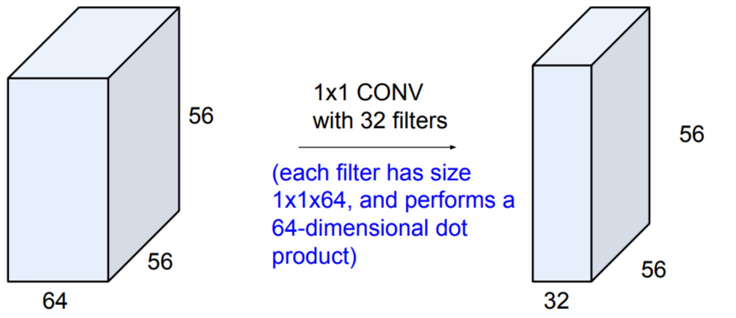
Inception + GoogLeNet (2015)– introduces parallel conv blocks (inception)

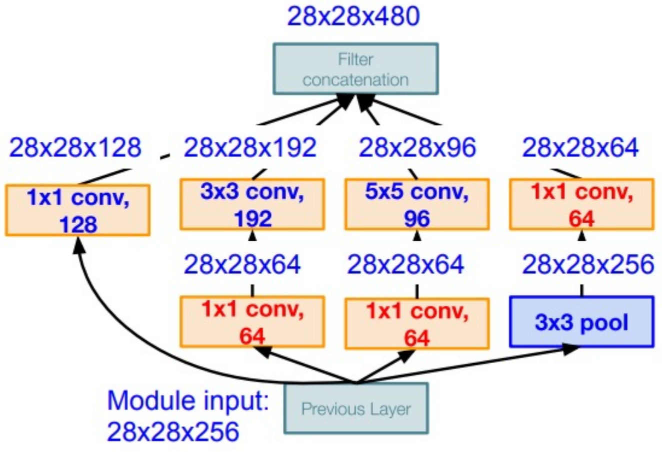
Cleverly uses 1x1 convolutions



Naive Inception module

Preserves spatial dimensions, but reduces depth! Feature maps (depth) are projected to lower dimension





Inception module with dimension reduction

From: http://cs231n.stanford.edu

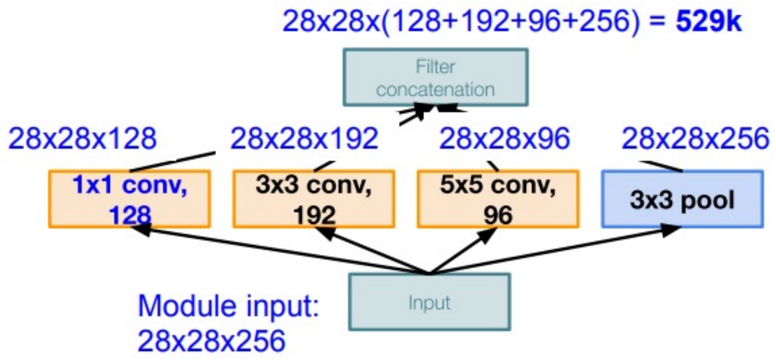
From: http://cs231n.stanford.edu

GoogLeNet (optional)

Inception + GoogLeNet (2015)– introduces parallel c

Cleverly

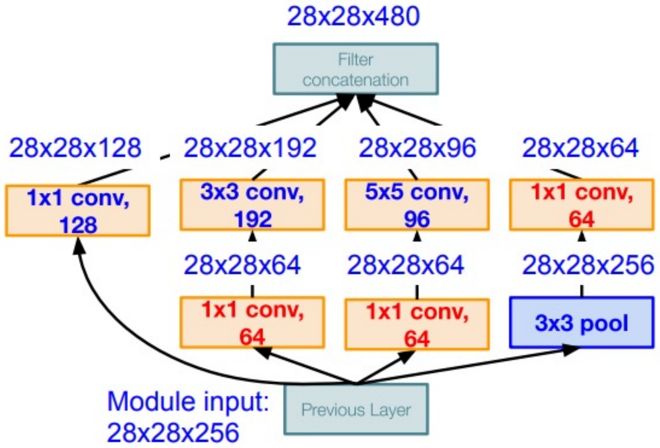
Conv Ops:
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x256
[5x5 conv, 96] 28x28x96x5x5x256
Total: 854M ops



Naive Inception module

Preserves spatial dimensions, but reduces depth! Feature maps (depth) are projected to lower dimension

Conv Ops:
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256
Total: 358M ops



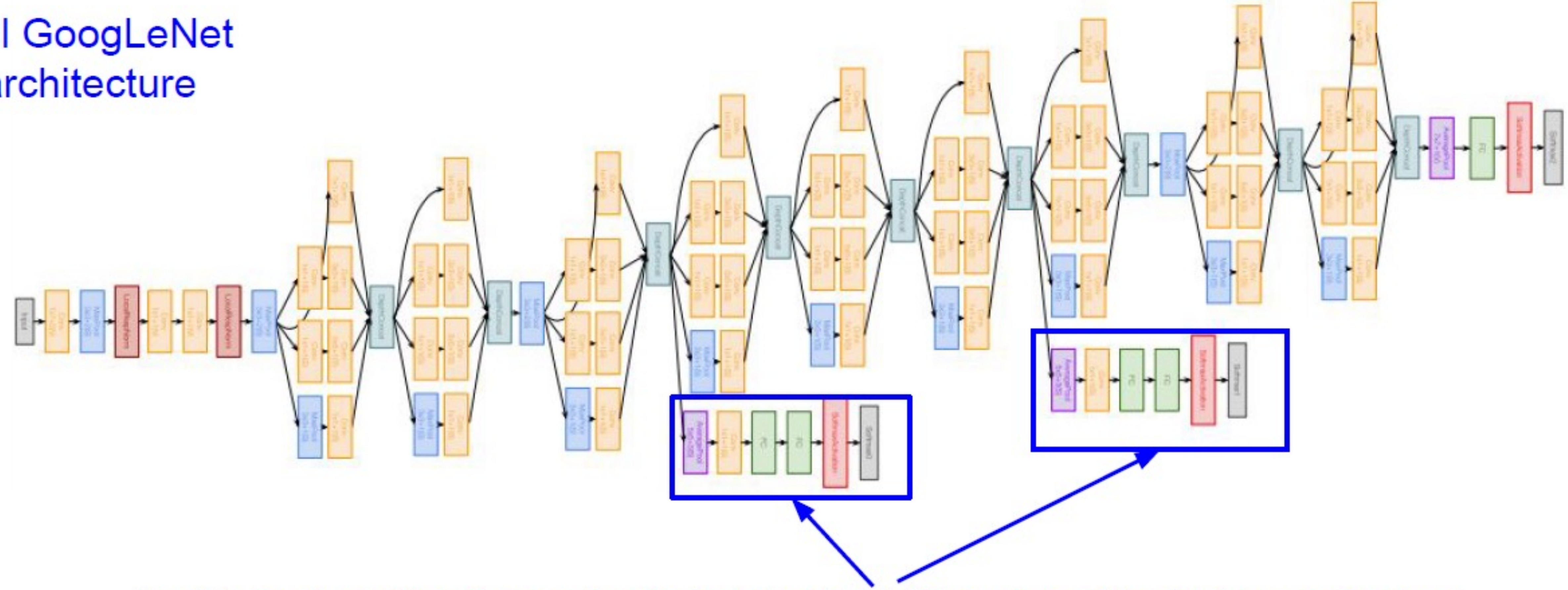
Inception module with dimension reduction

From: http://cs231n.stanford.edu

From: http://cs231n.stanford.edu

GoogLeNet (optional)

Full GoogLeNet architecture



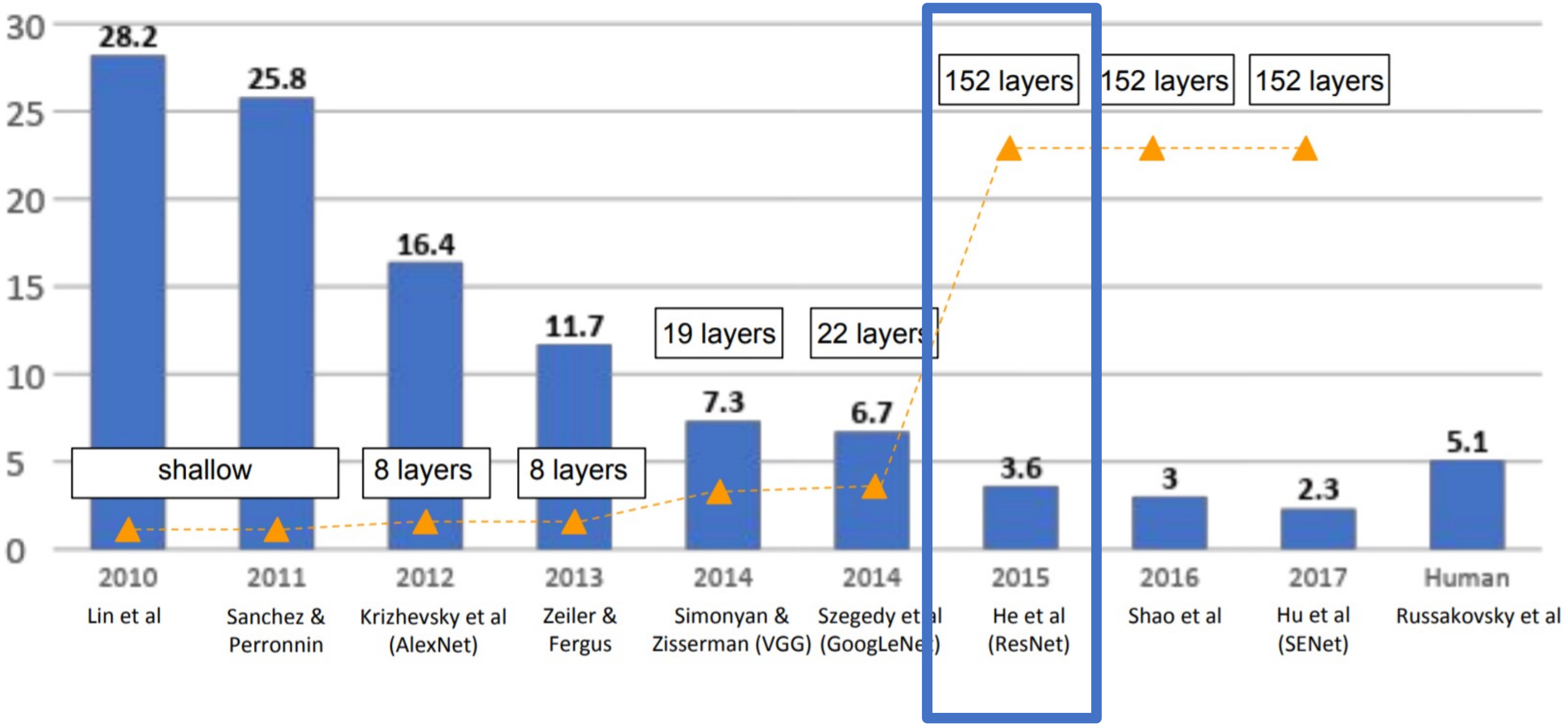
Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

From: http://cs231n.stanford.edu

An overview on the most famous architectures

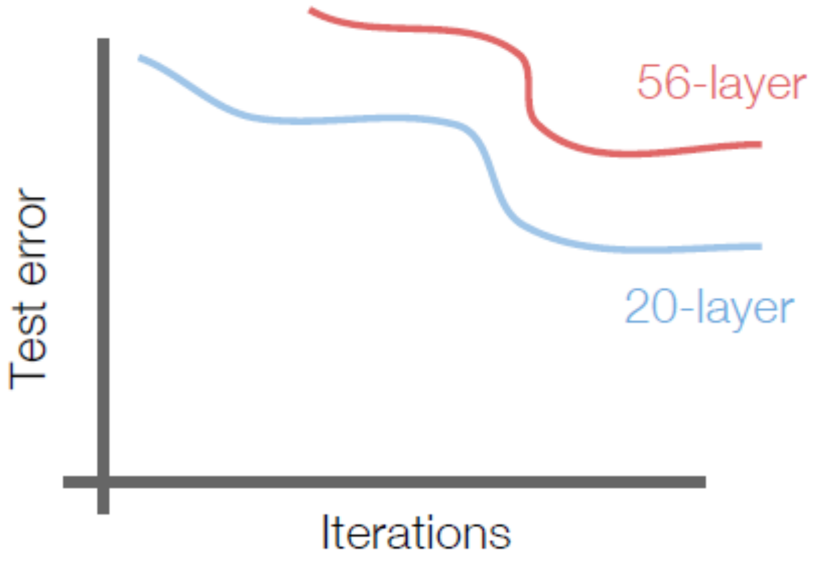
Imagenet – visual recognition challenge with 1000 classes.

Winners:



ResNet

Why not stacking more and more layers?

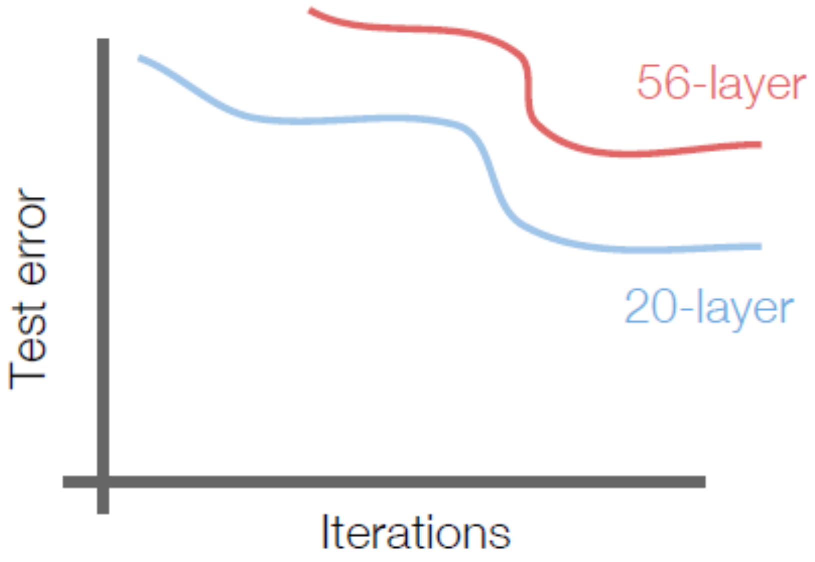


From: http://cs231n.stanford.edu

ResNet

Why not stacking more and more layers?

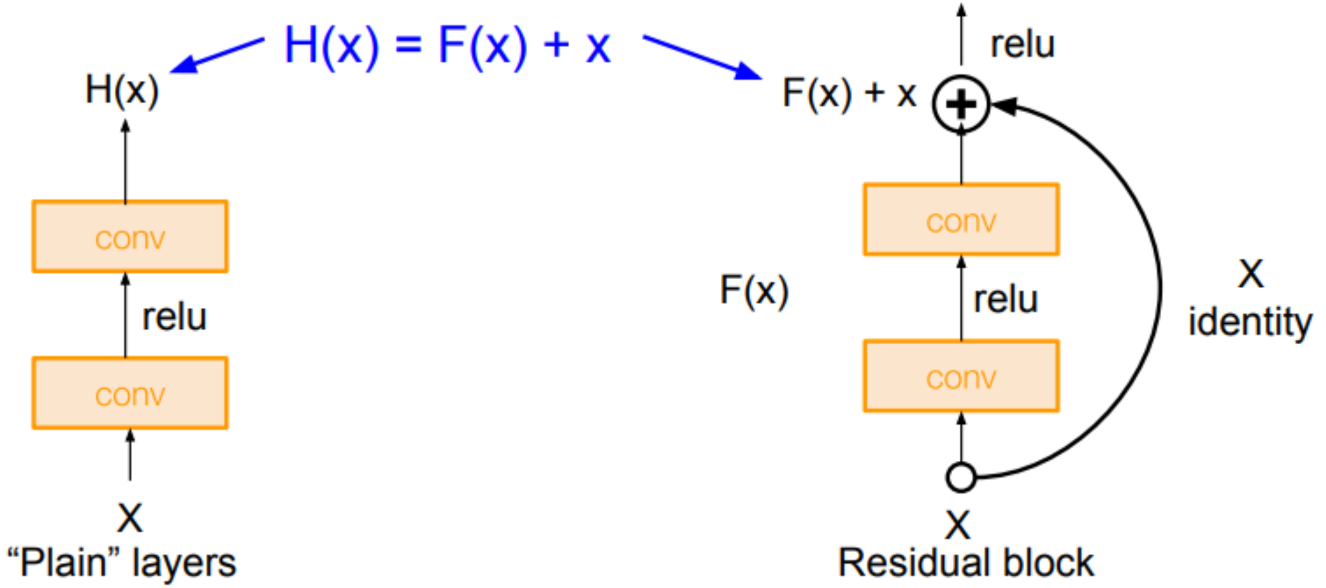
Even with all of the 'tricks' from some point onward, stacking more layers make the training really hard!



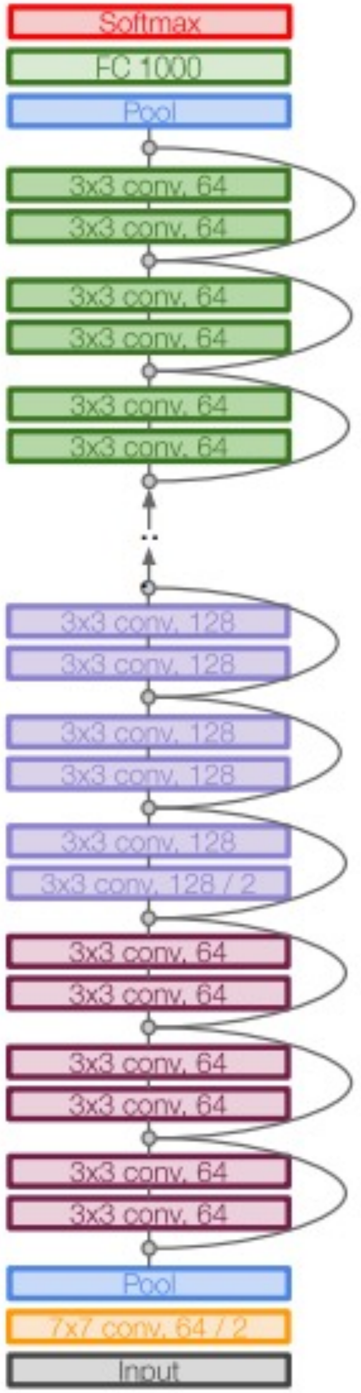
From: http://cs231n.stanford.edu

ResNet

[ResNet](#) (2015) – very deep model (152 layer) with shortcut connections



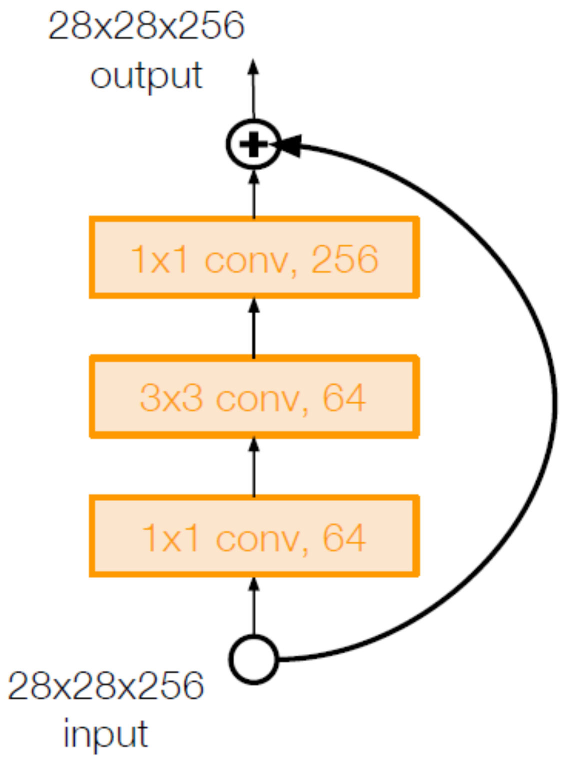
Use layers to fit residual F(x) = H(x) - x instead of H(x) directly



From: http://cs231n.stanford.edu

From: http://cs231n.stanfo

ResNet



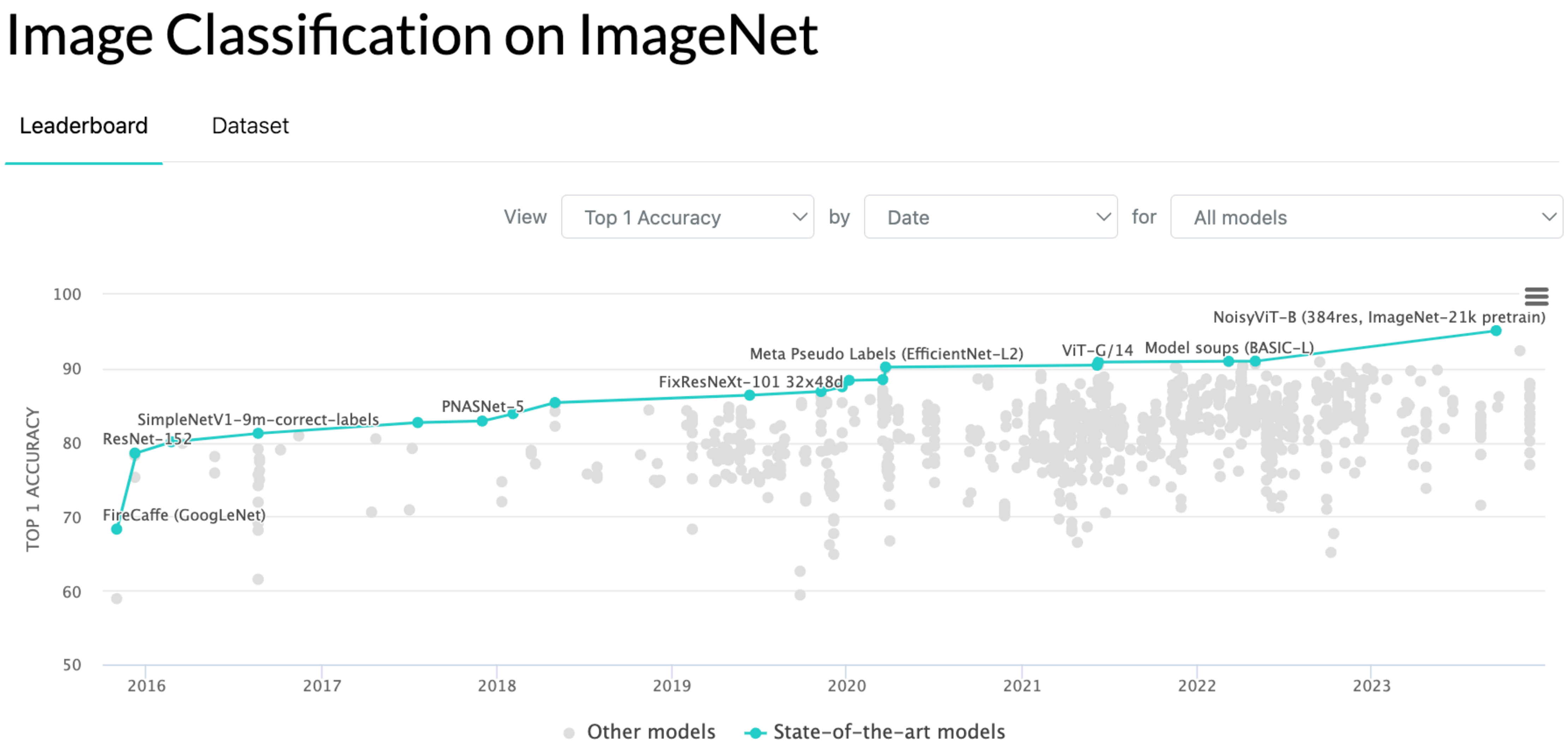
For ResNet with more than 50 layers

From: http://cs231n.stanford.edu

ResNet training:

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

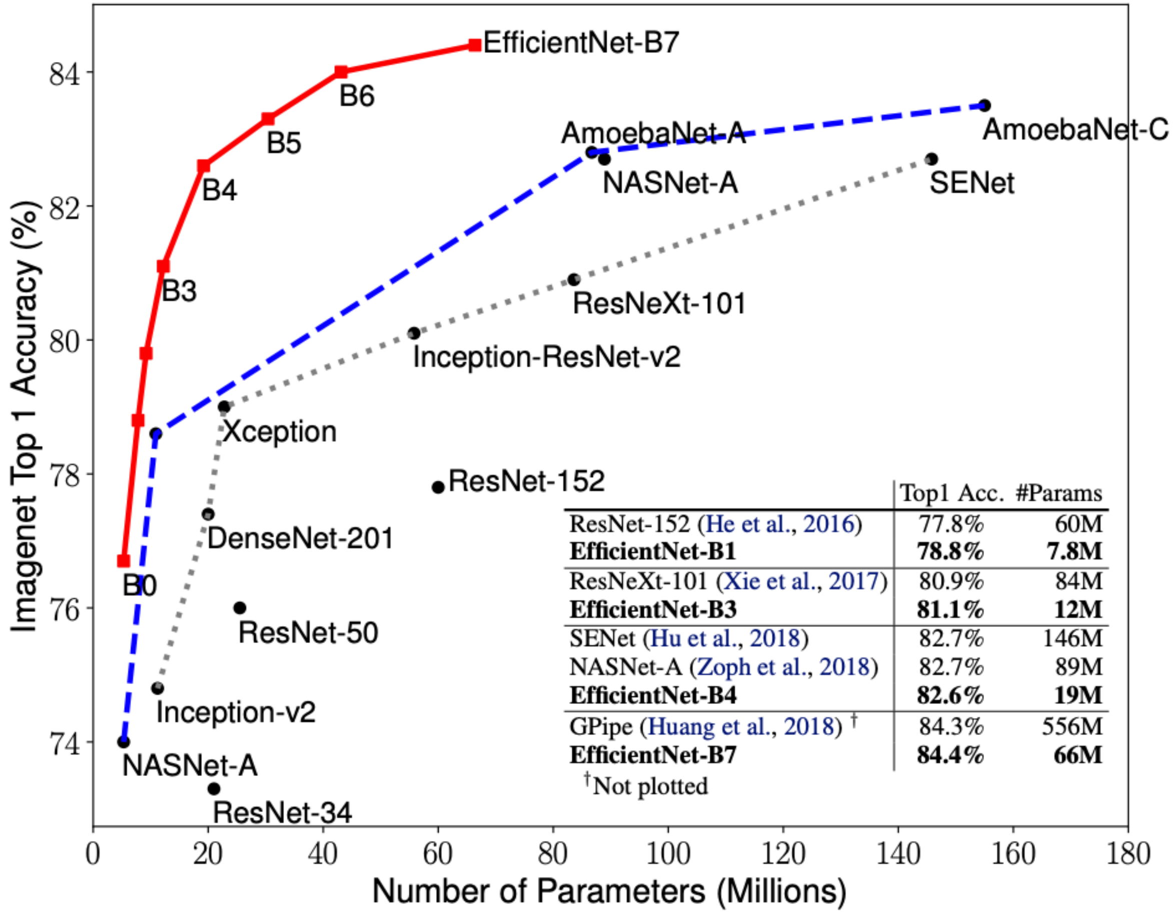
State of the art is always on the move...



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

Is not all about accuracy… EfficientNet

Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (pp. 6105-6114). PMLR.



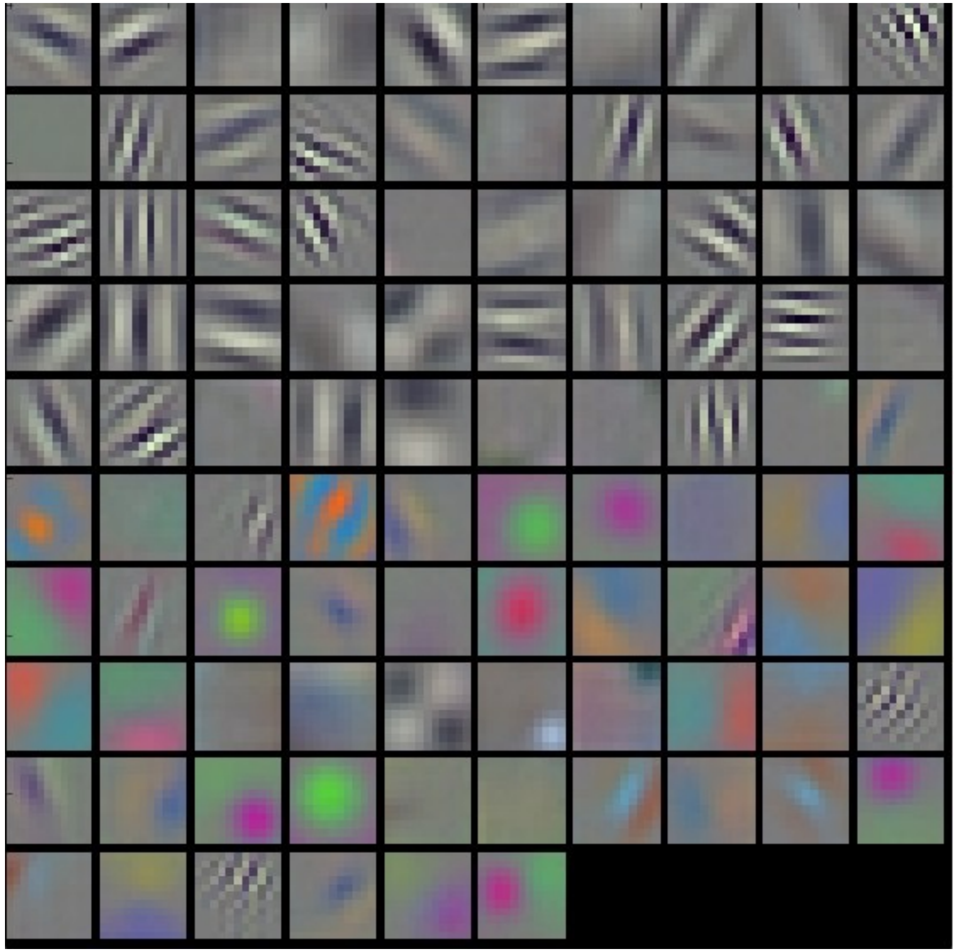
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

† Not plotted

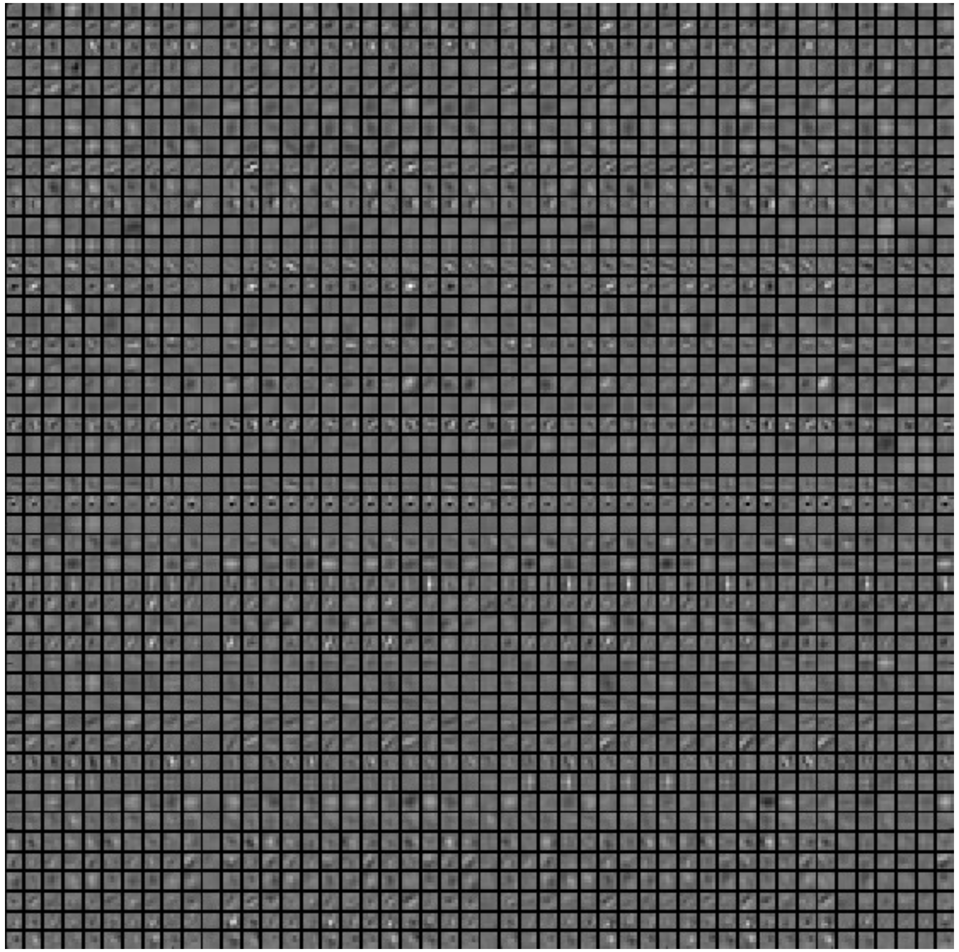
Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.4% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

What do CCN see? (Optional)

What do CNN see? Visualize the layers



AlexNet 1CONV layer

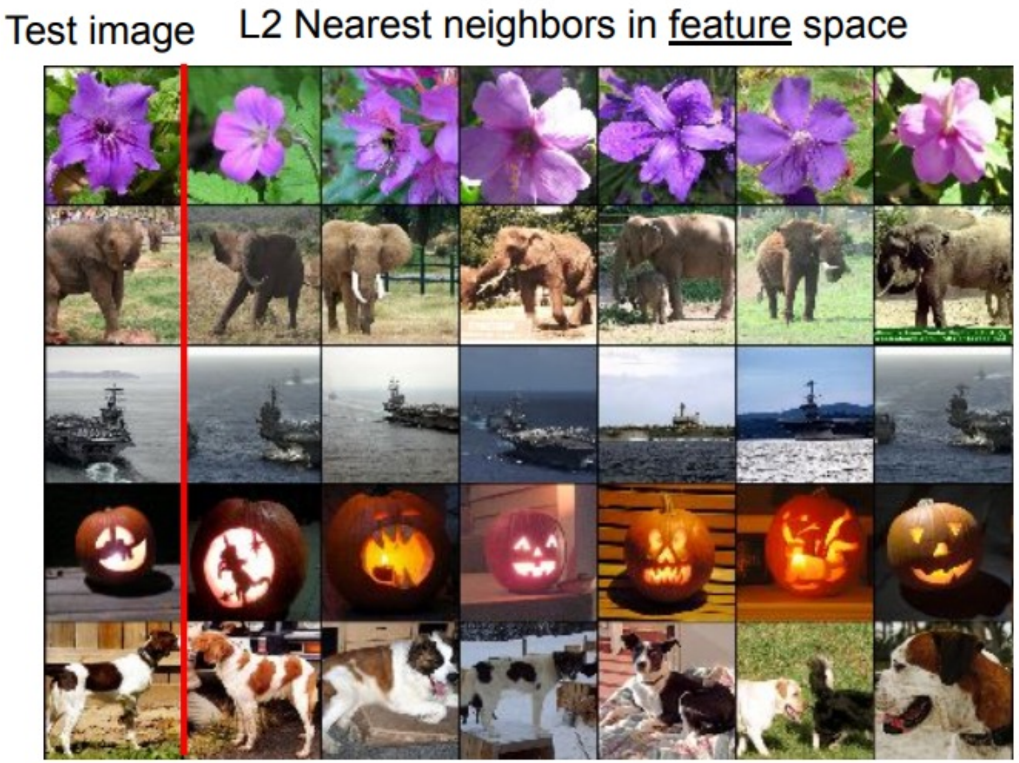


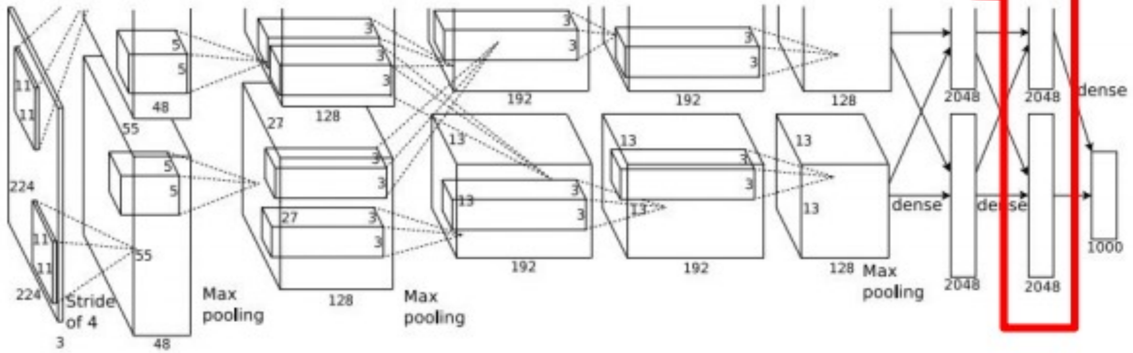
AlexNet 2CONV layer

From: http://cs231n.stanford.edu

What do CNN see? Embedding space for features

We can **consider k-nearest neighbors in embedding space for last FC layer:**





Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

From: http://cs231n.stanford.edu

What do CNN see? Embedding space for features

We can **plot final FC embedding layer by means of dimensionality reduction**, e.g. tSNE (more powerful than PCA) or UMAP

Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

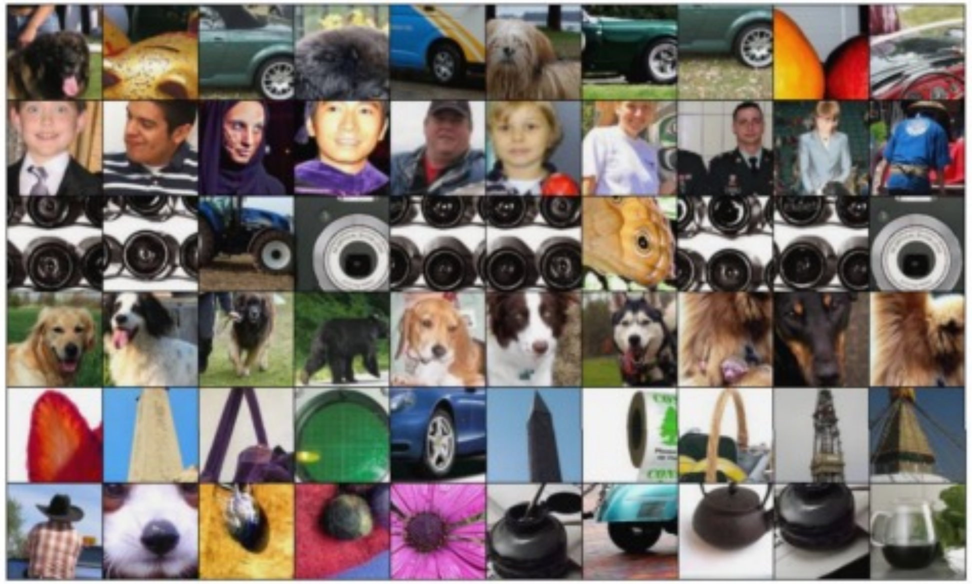
From: http://cs231n.stanford.edu

What do CNN see? Maximally activating neuros

We can **compute maximally activating patches.**

Run many images through the network, record values of chosen channel (e.g. channel 17/128 in conv5).

Visualize image patches that correspond to maximal activations.

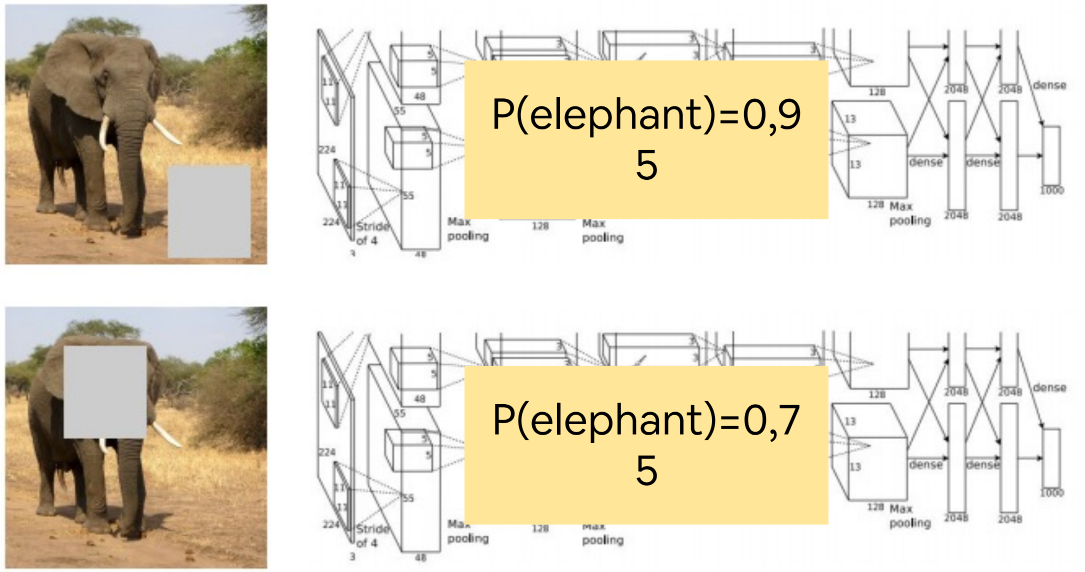


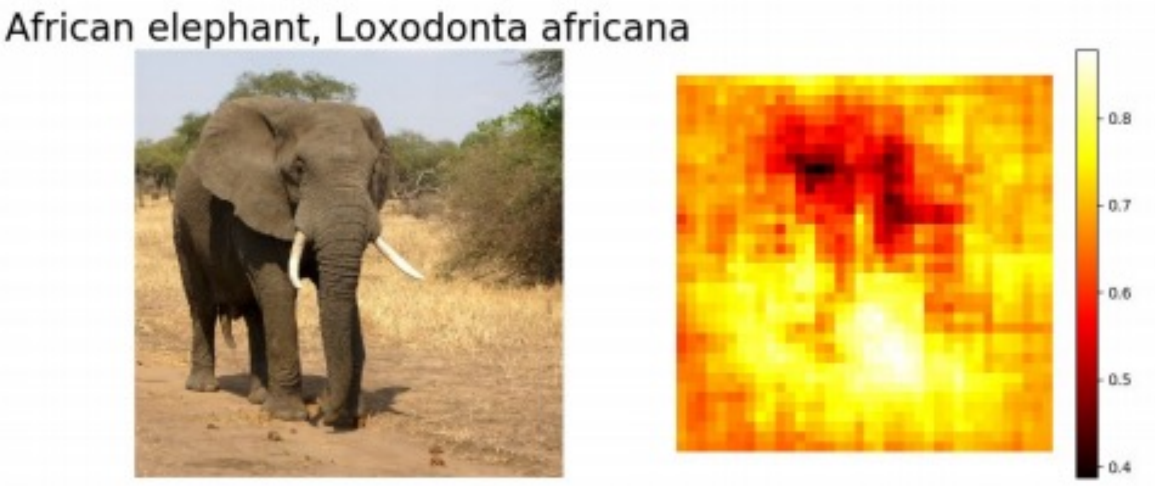
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015 Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015;

From: http://cs231n.stanford.edu

What do CNN see? Most relevant pixels

Saliency maps, e.g. by occlusion:





Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

From: http://cs231n.stanford.edu

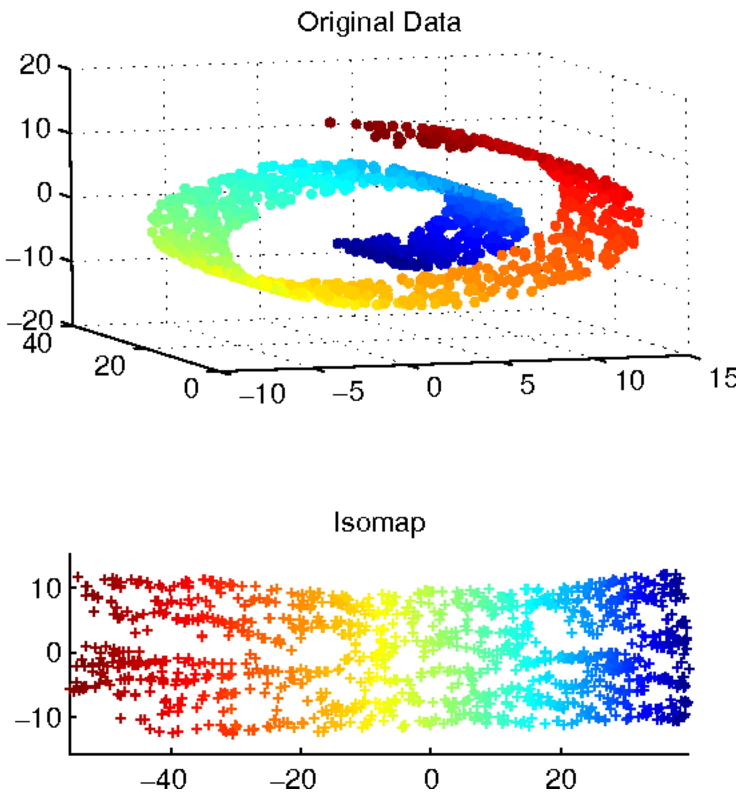
Unsupervised Learning in DL: Autoencoders

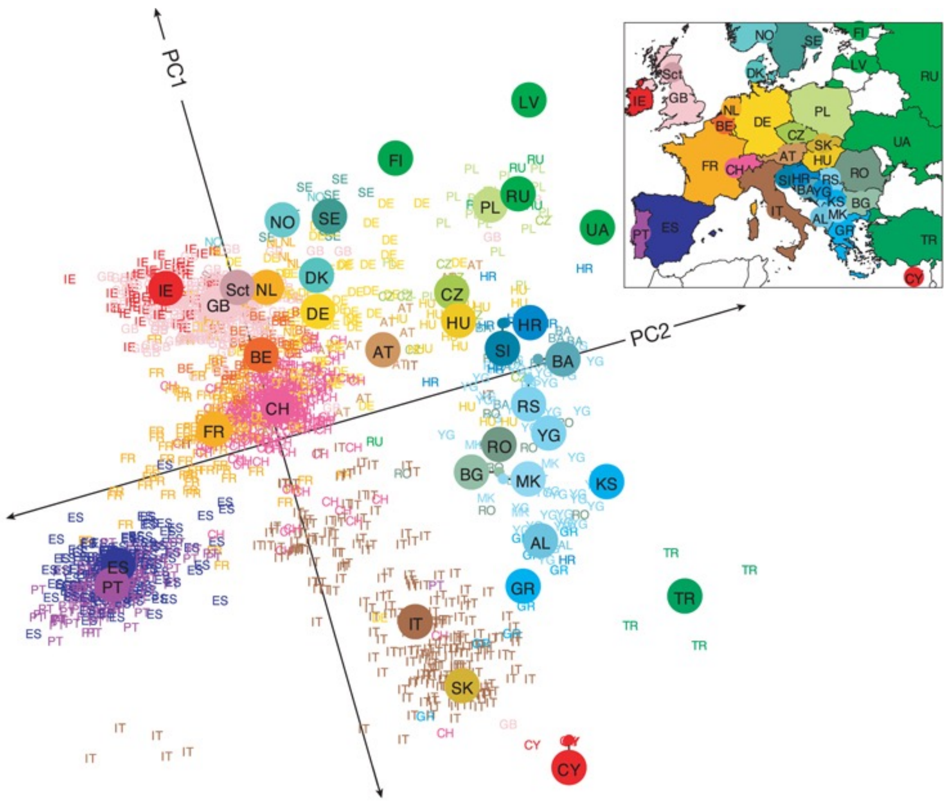
Unsupervised Learning Tasks

- Clustering
- Dimensionality Reduction/Learning latent representations (Representation learning)
- Anomaly Detection
- Data Generation

Dimensionality Reduction/Learning latent representations

Hypothesis: in high-dimensional data sets, the data nearly always lie on (or close to) a much lower-dimensional, smoothly curved manifold

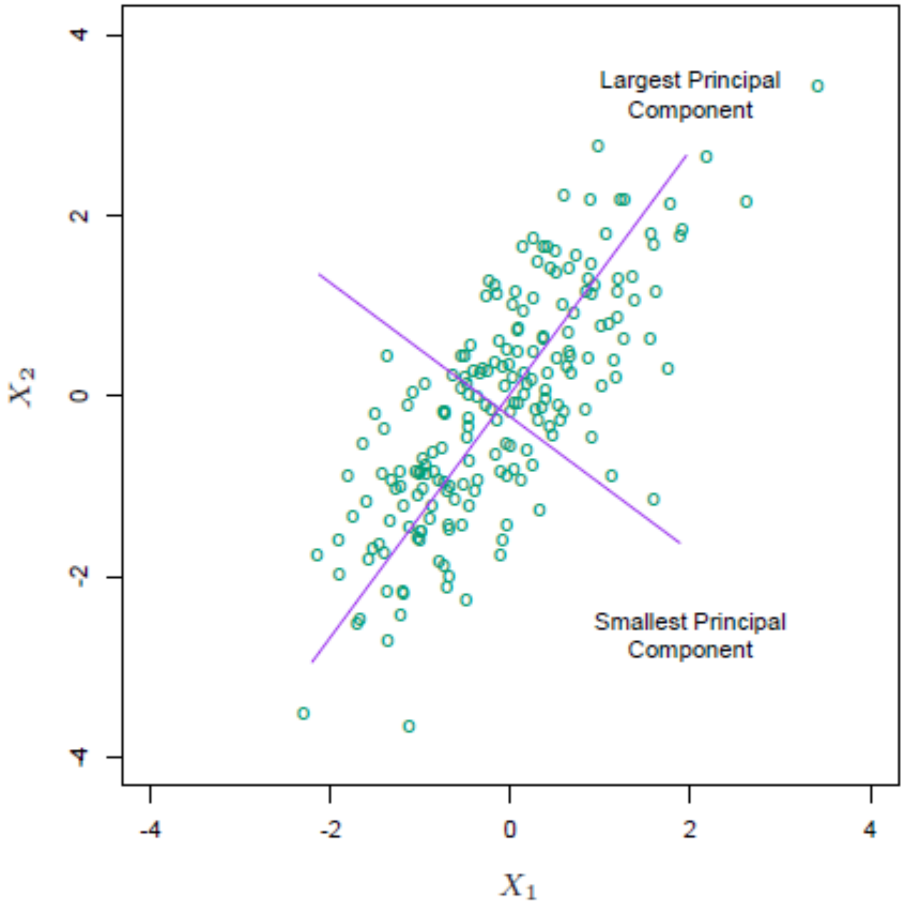


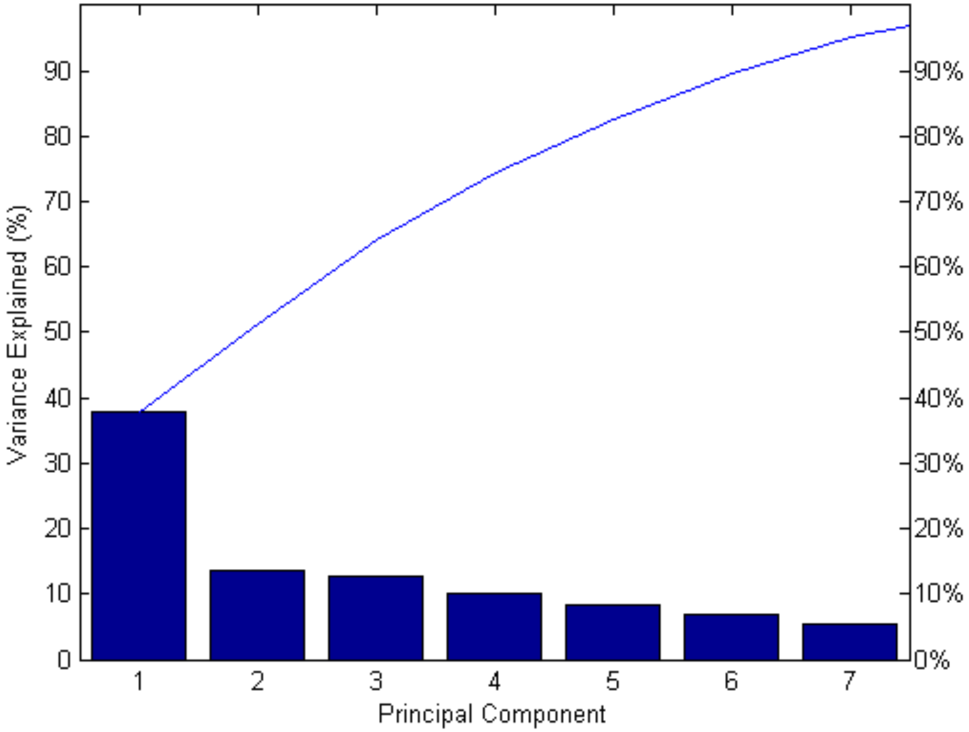


Genes mirror geography within Europe – Nature 2008

Dimensionality Reduction/Learning latent representations

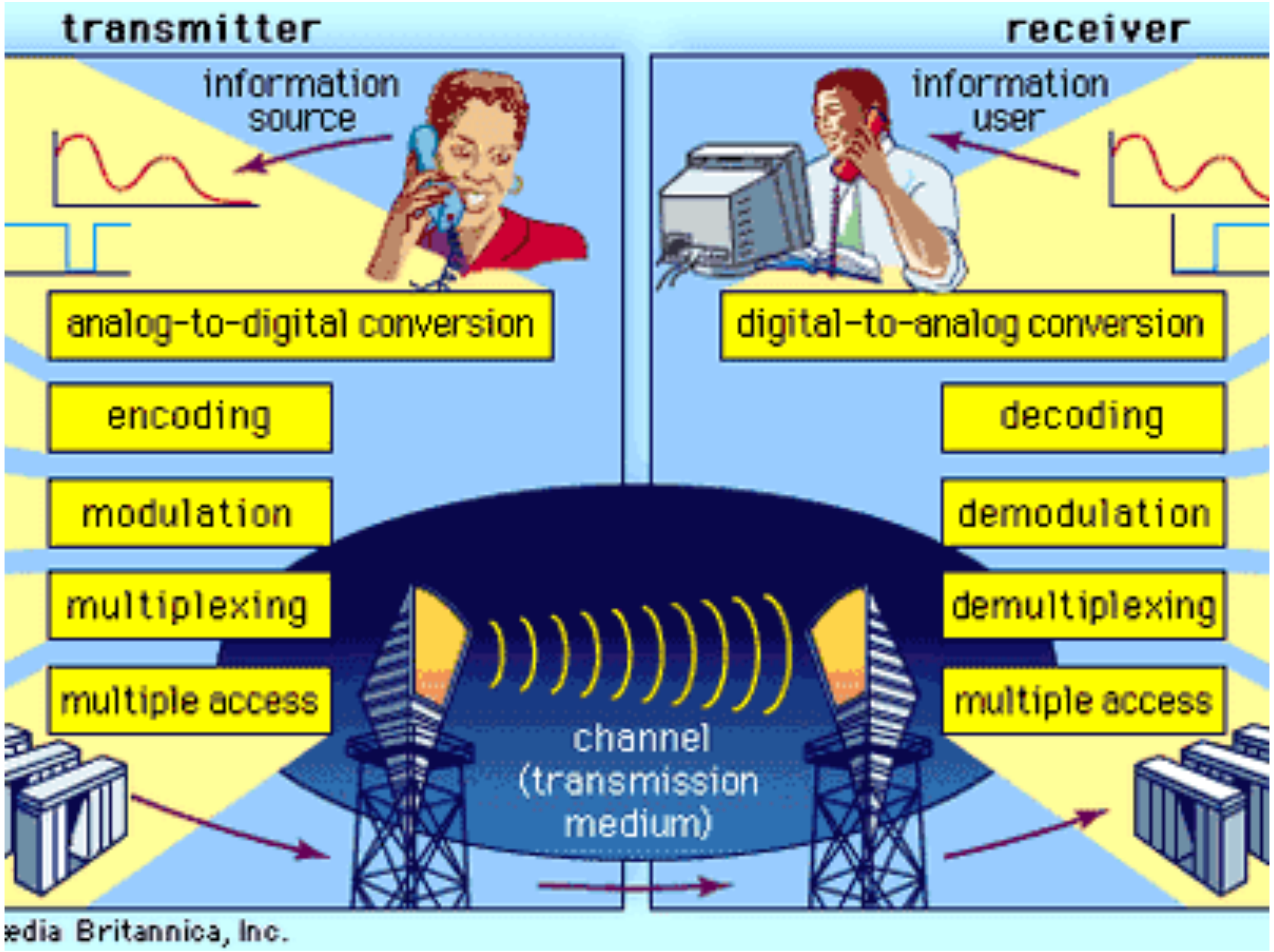
Simplest approach: Principal Component Analysis



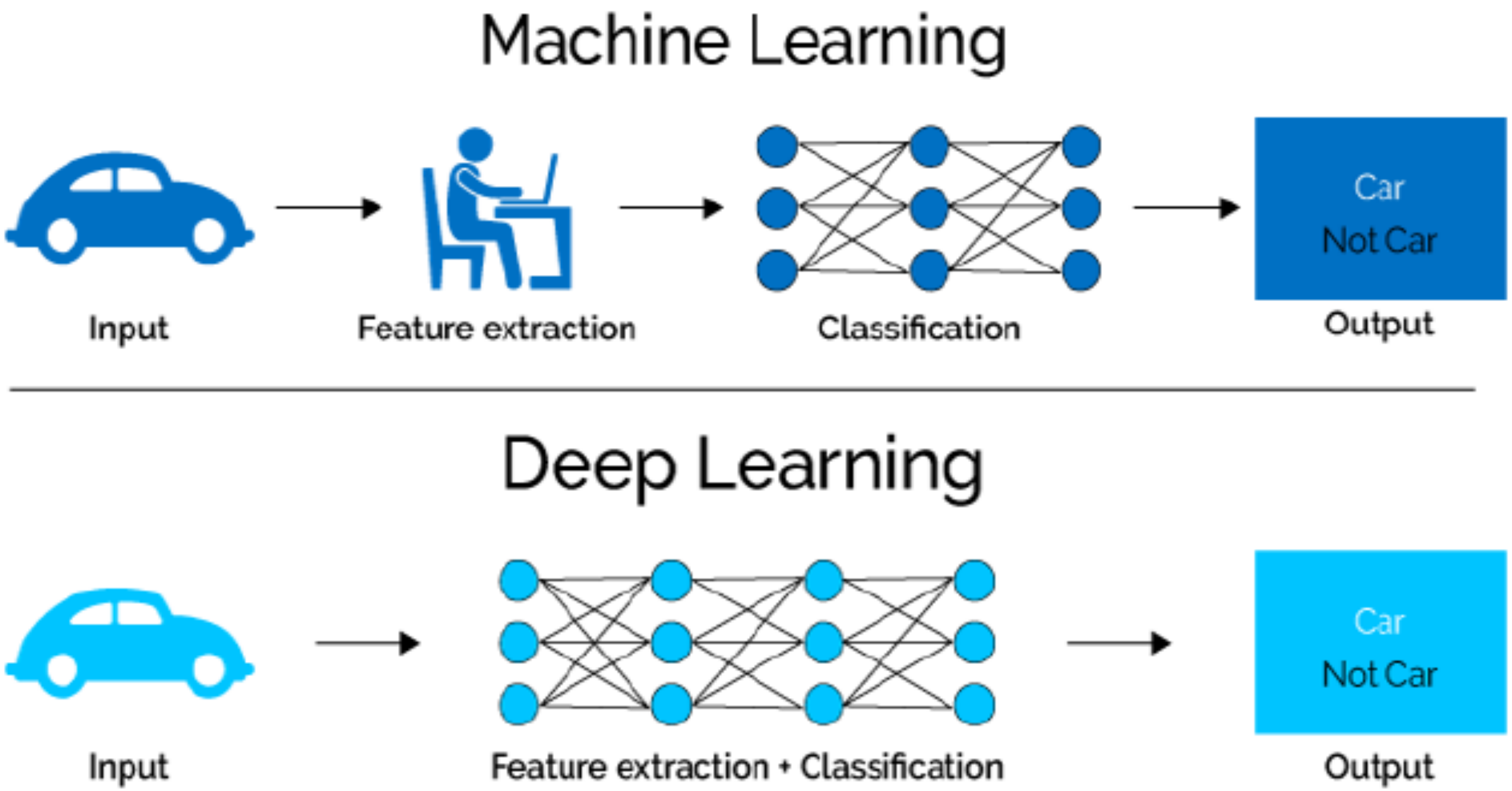


The problem of finding meaningful and minimal representation is recurring in engineering...

- In Telecommunication we have the encoding and decoding of a signal before and after the transmission



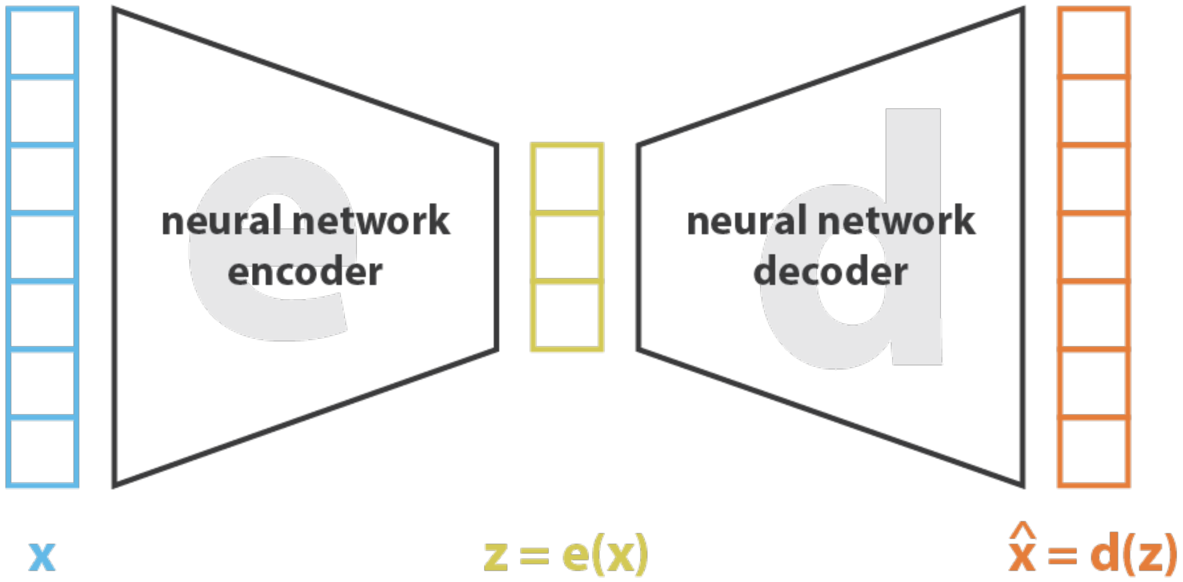
This is a recurring problem, with different setting



L. Fridman MIT Deep Learning https://deeplearning.mit.edu/

Autoencoders

- Deterministic models trained using error backpropagation
- Input and Output are the same data: we force a network to be able to reconstruct such data with the limitation of having a ‘bottleneck’ (code) of limited size

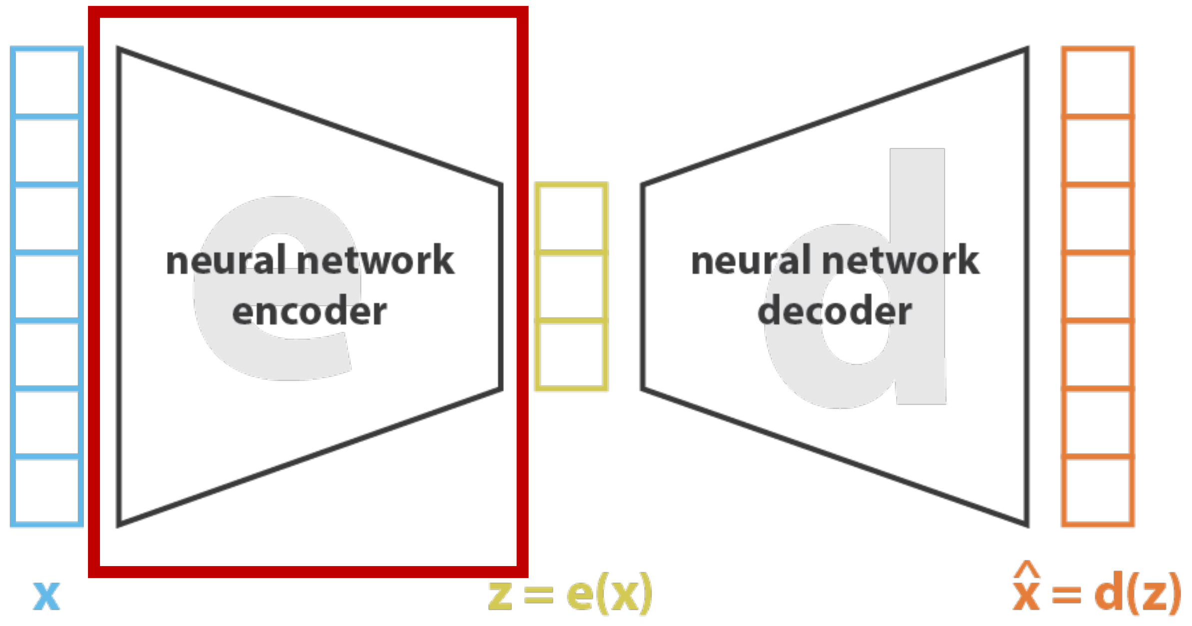


$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Autoencoders

- Deterministic models trained using error backpropagation
- Input and Output are the same data: we force a network to be able to reconstruct such data with the limitation of having a ‘bottleneck’ (code) of limited size

The encoder provides a low dimensional representation of the input

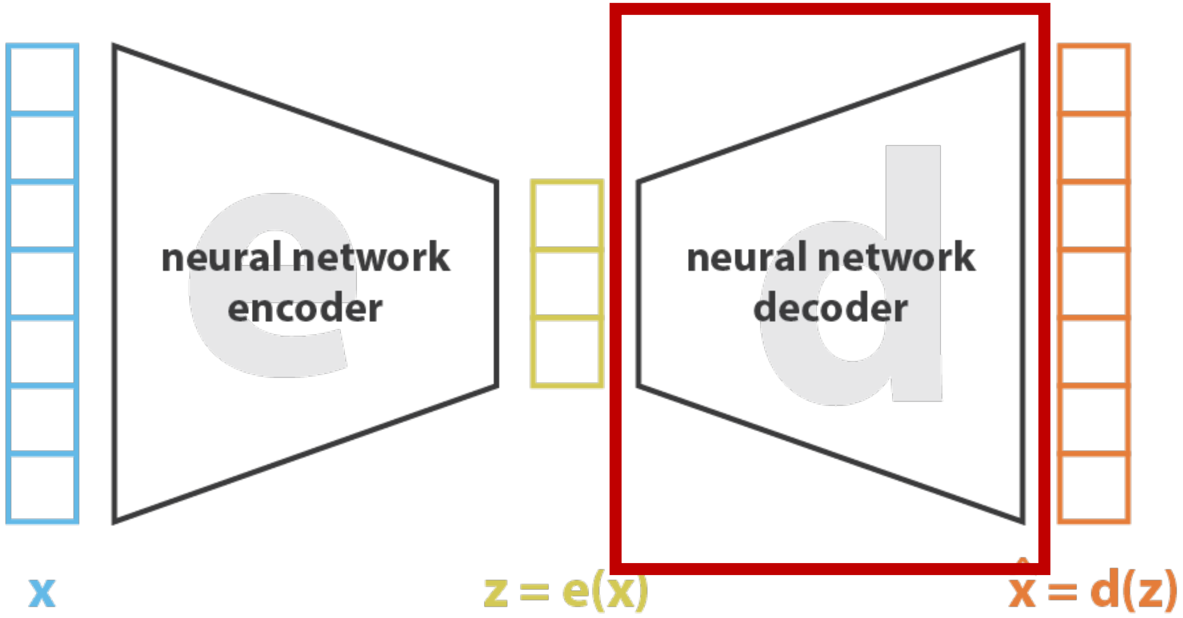


$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Autoencoders

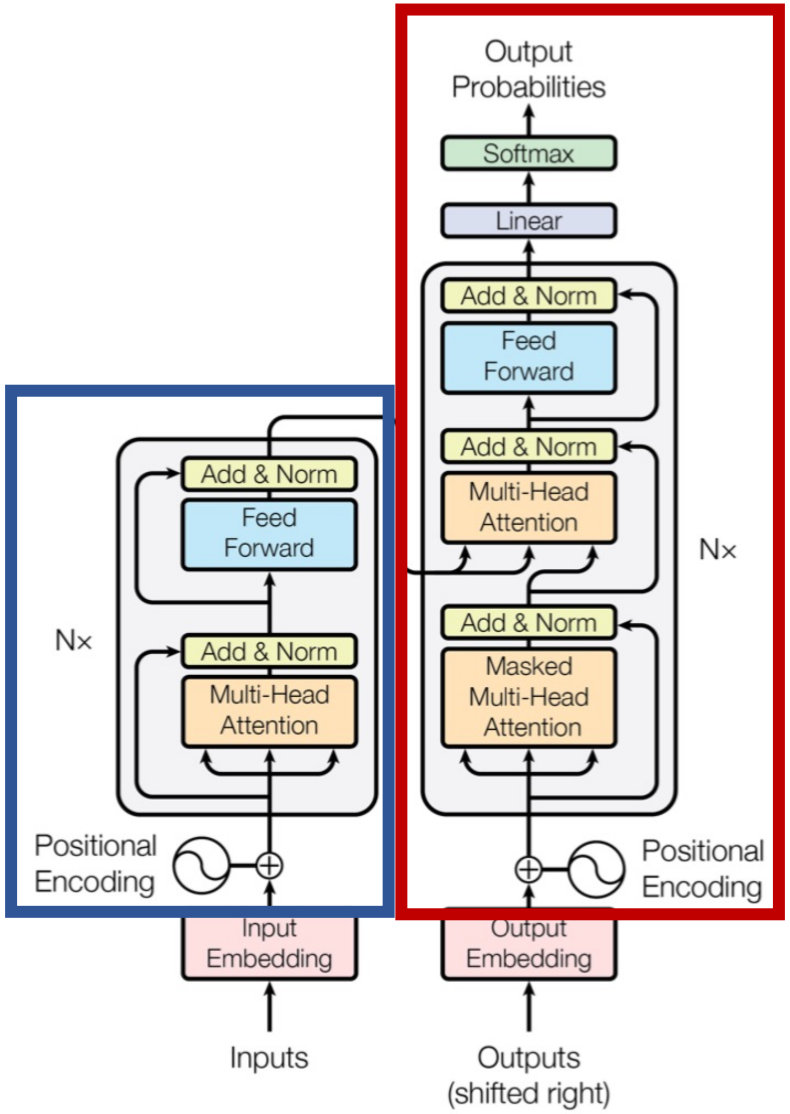
- Deterministic models trained using error backpropagation
- Input and Output are the same data: we force a network to be able to reconstruct such data with the limitation of having a ‘bottleneck’ (code) of limited size

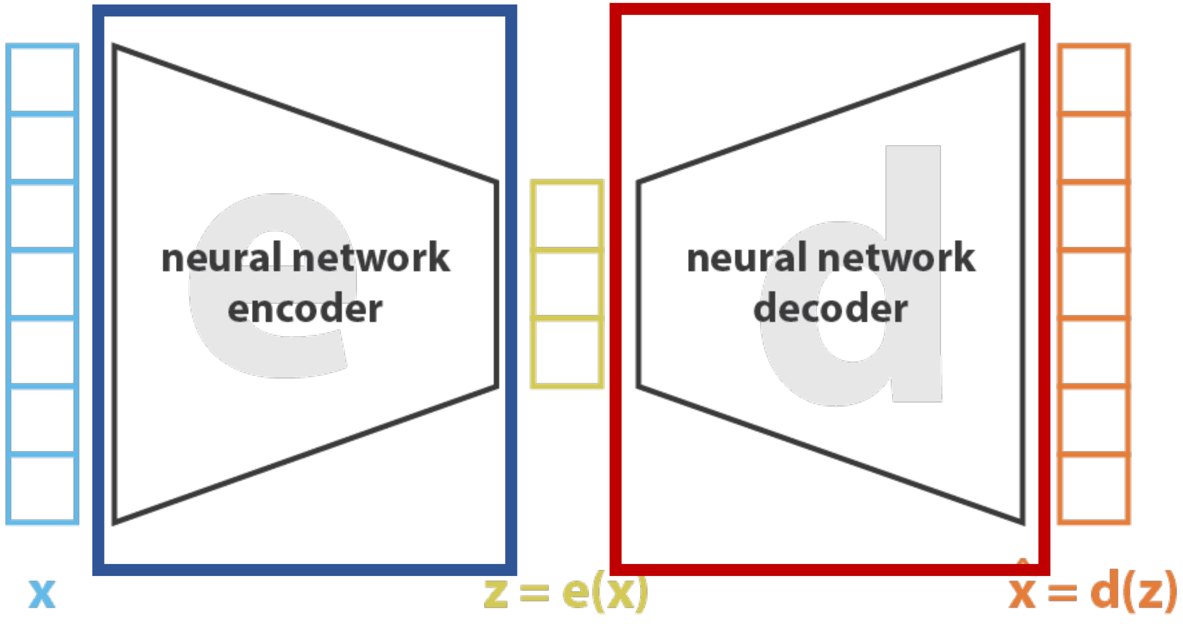
The decoder reconstructs the input from its compressed representation



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Autoencoders

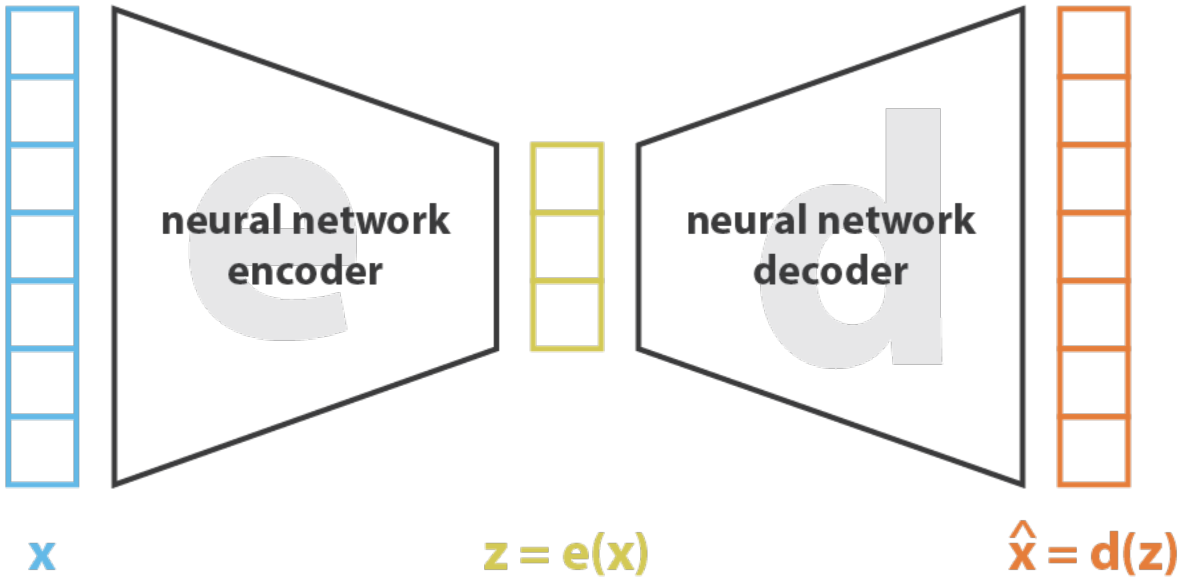




$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Autoencoders

- Encoder and decoder can have different structure, however we can regularize the network by imposing a symmetric architecture
- Typically, the number of hidden units is chosen to be lower than input units

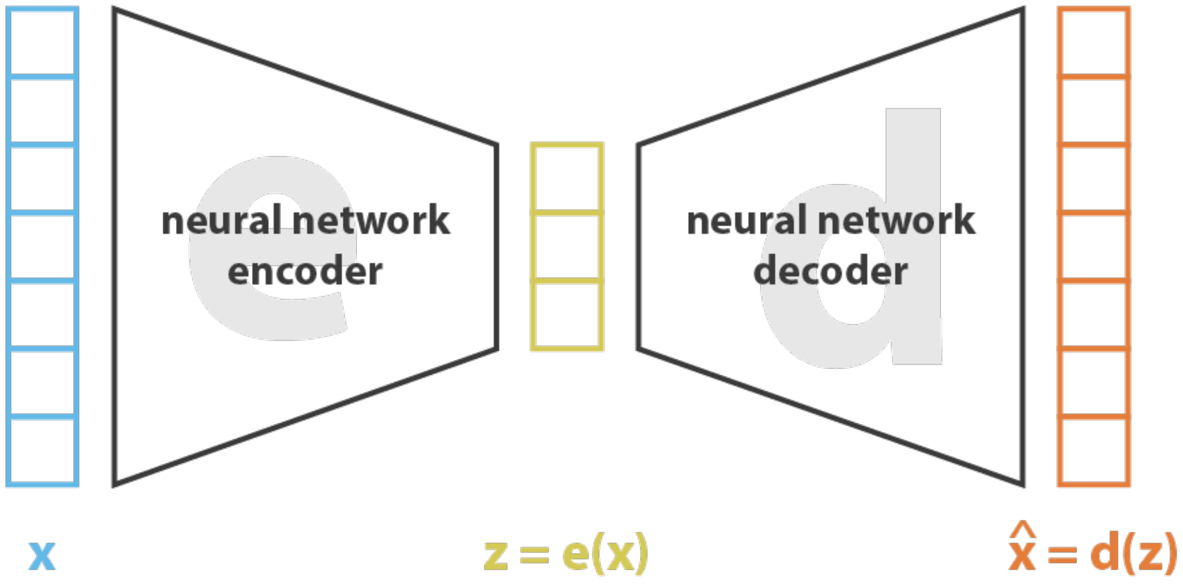


$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Autoencoders

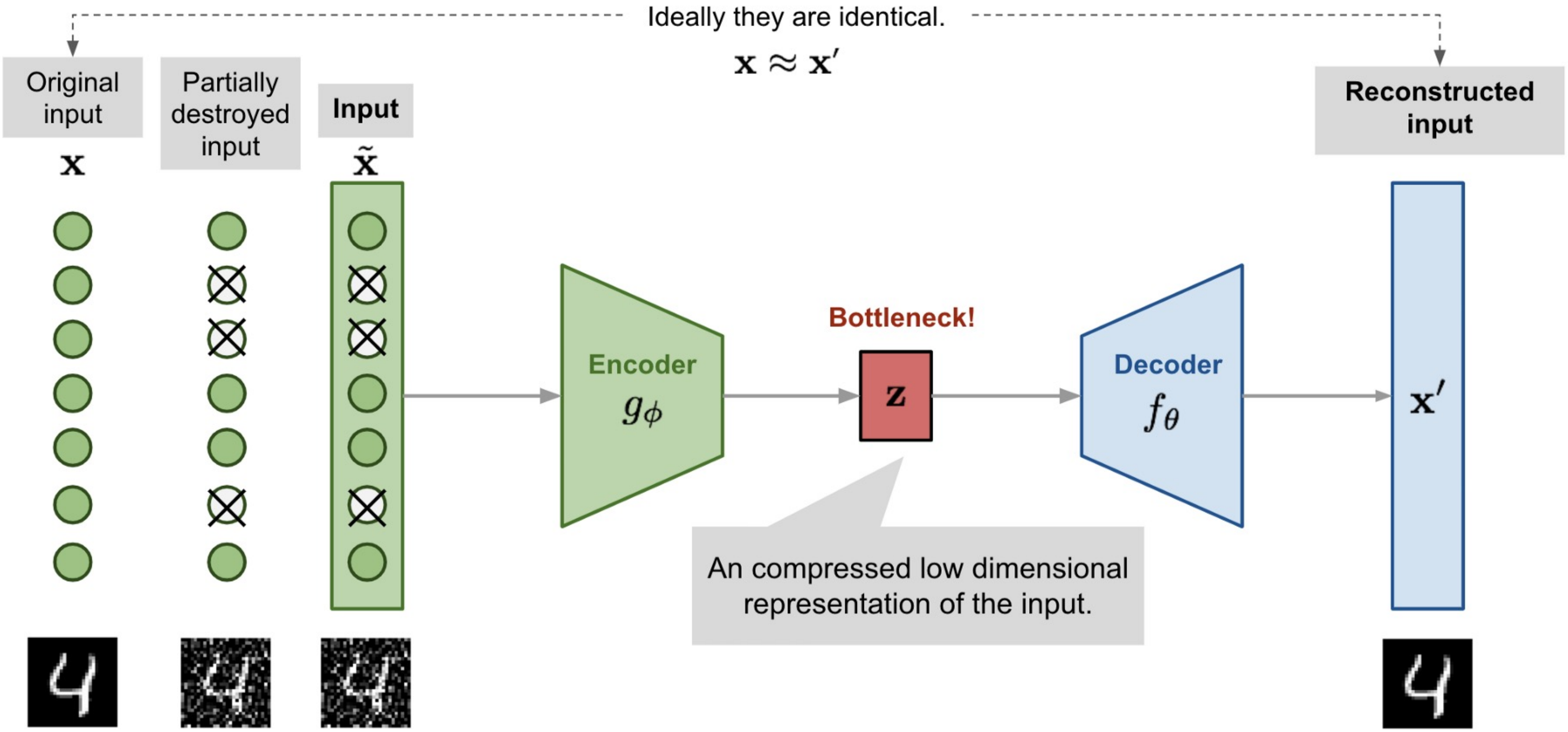
- With linear activations we can get PCA
- We can introduce regularizes to learn even more meaningful representations:

1. Sparse autoencoders (L1 penalty on hidden activations)
2. Denoising autoencoders

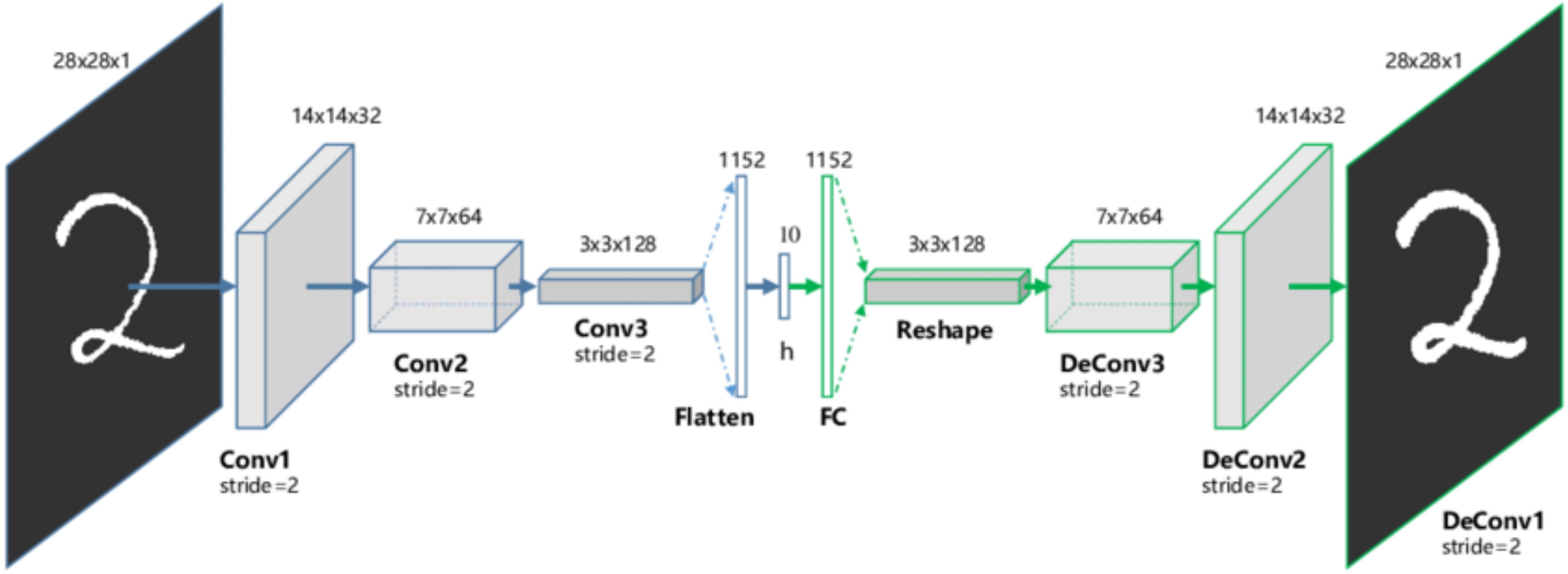


$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

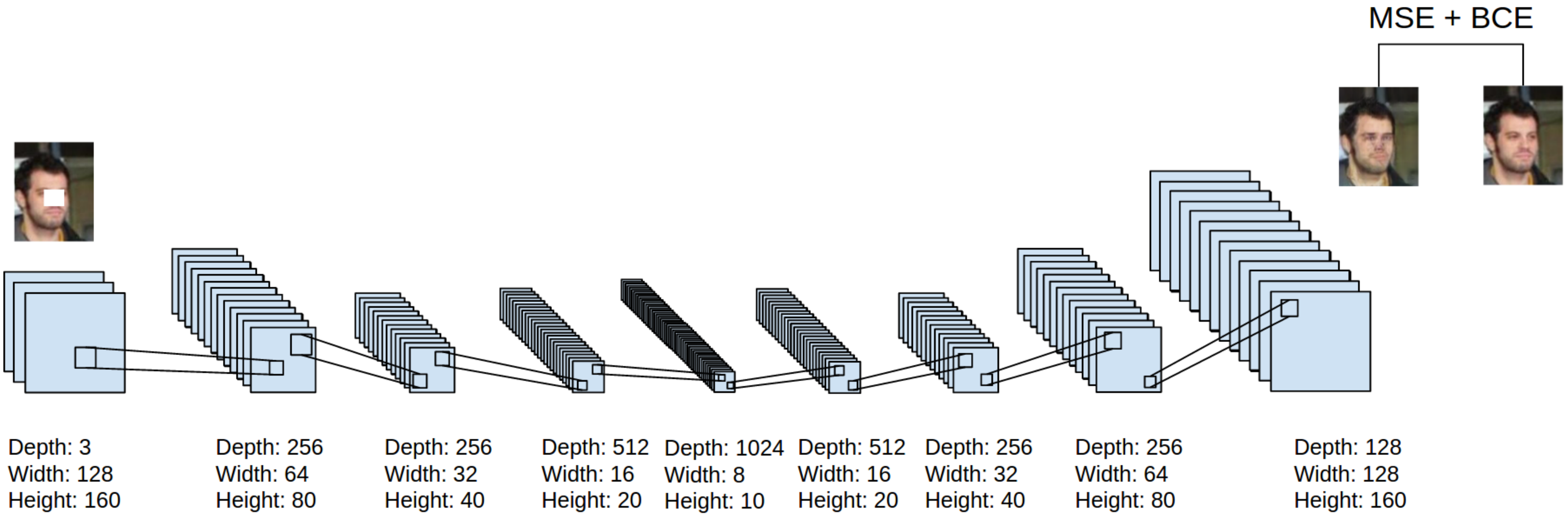
Denoising Autoencoders



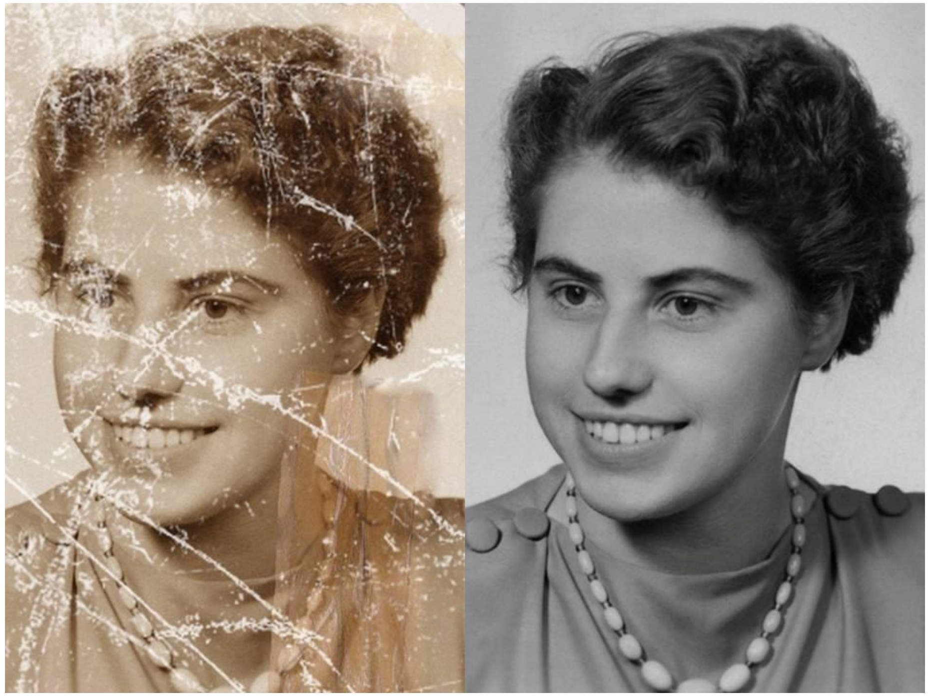
… we can still use convolutions…

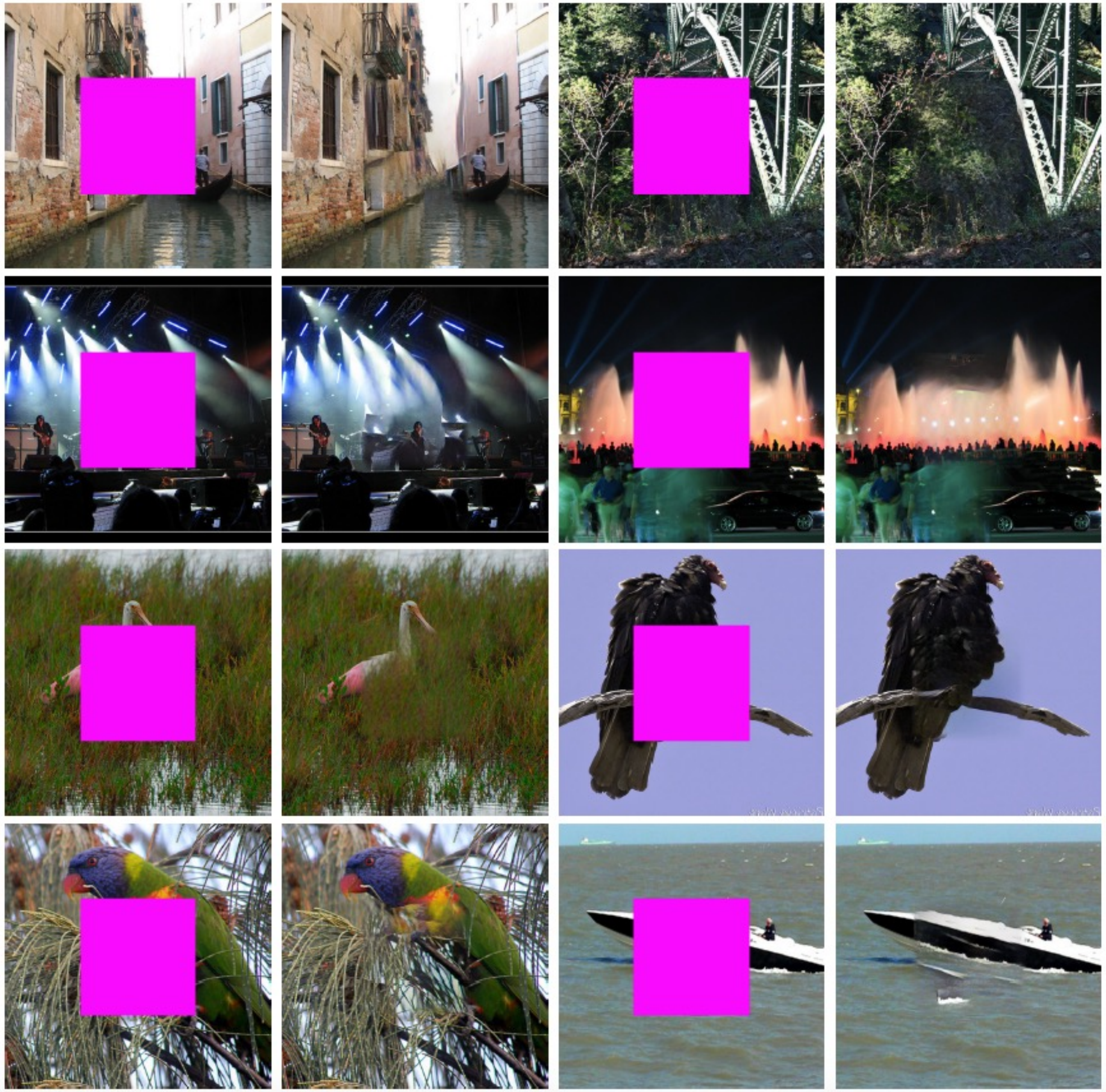


Neural inpainting

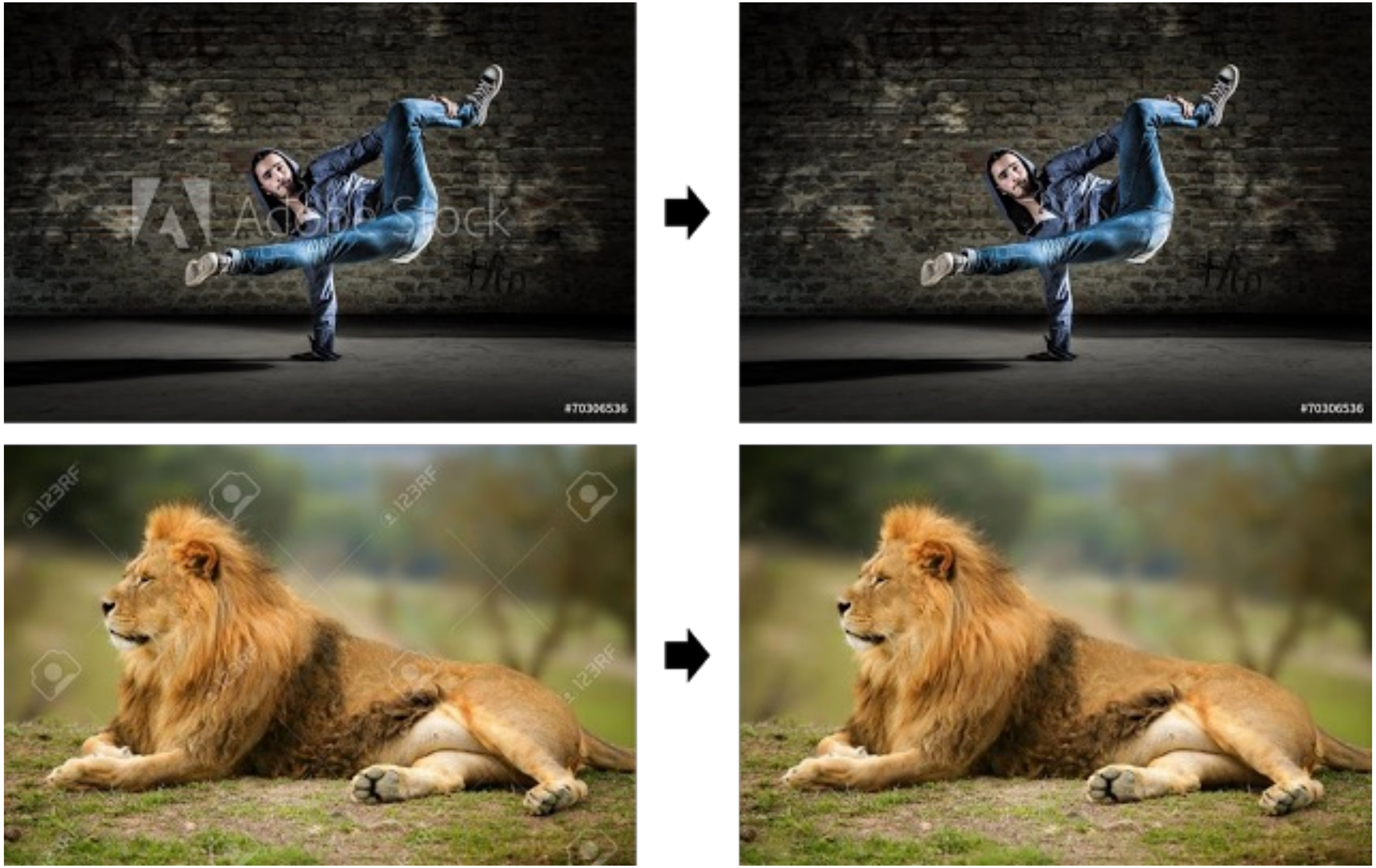


Neural inpainting





Neural inpainting



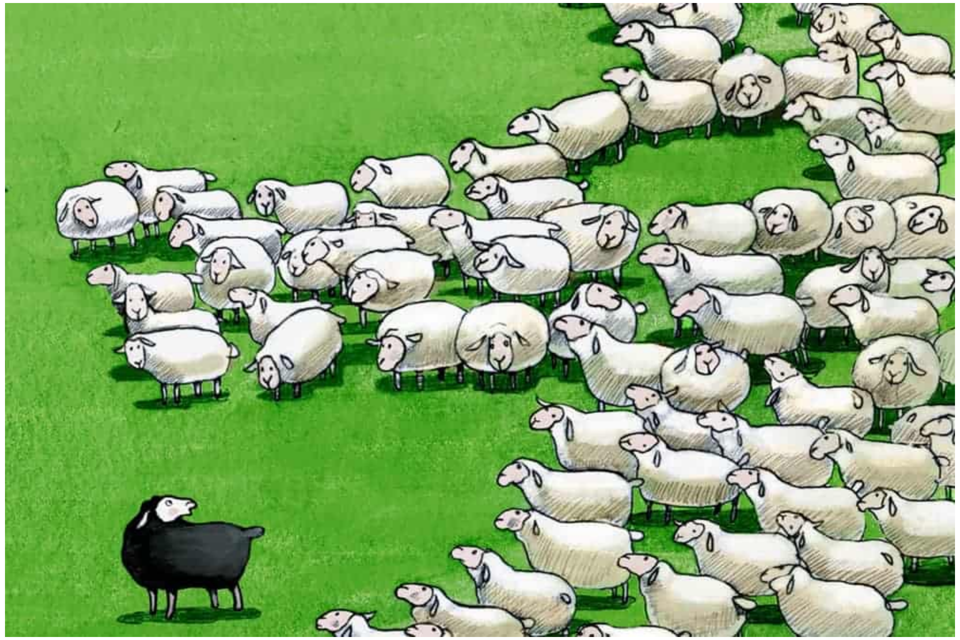
Unsupervised Learning Tasks

- Clustering
- Dimensionality Reduction/Learning latent representations (Representation learning)
- Anomaly Detection
- Data Generation

Anomaly Detection

What is an anomaly/outlier?

‘An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism’ (Hawkins definition [1])



D. M. Hawkins, Identification of outliers, vol. 11., Springer, 1980.

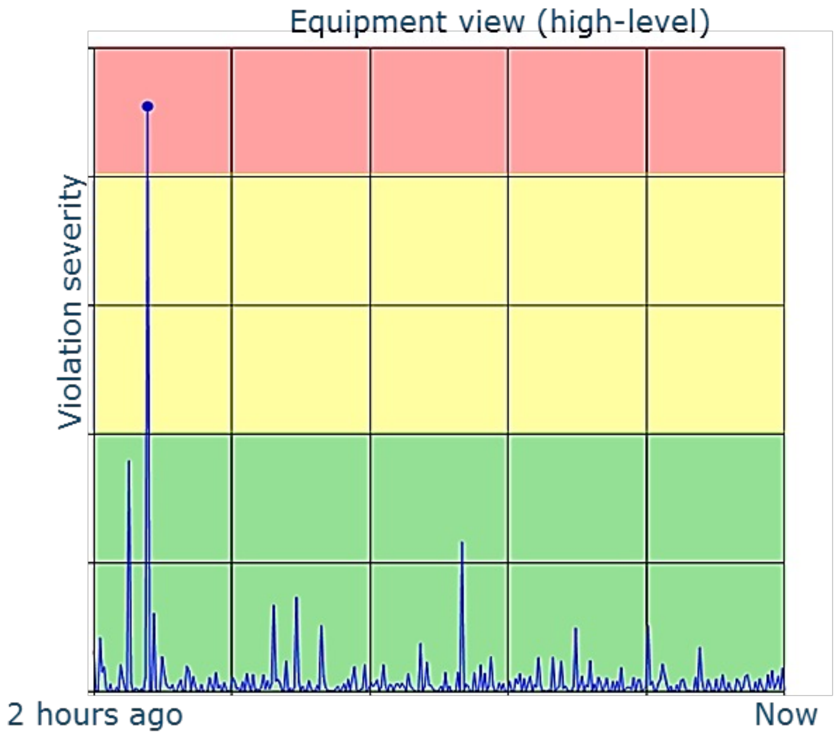
Multivariate Anomaly Detection

Such approaches allow us to provide ‘anomaly scores’: unique quantitative indicators able to represent the degree of ‘outlierness’ of complex systems with many variables

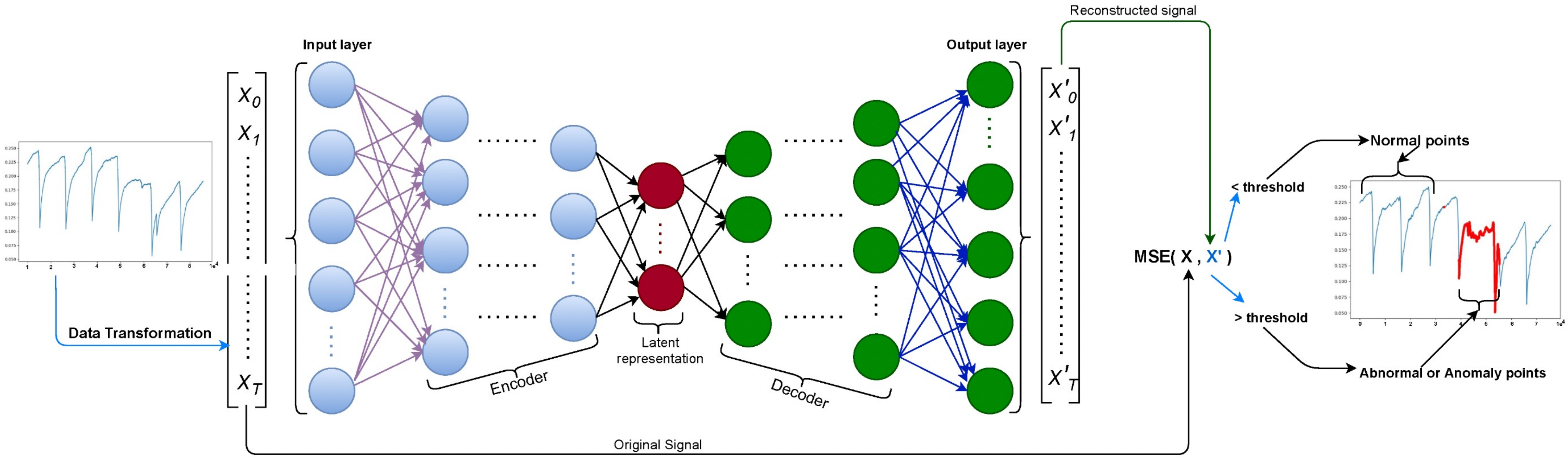
Many approaches:
- Density-based methods (e.g. LOF, DBSCAN)
- Distance-based methods (e.g. ORCA)
- Clustering-based methods (e.g. CBLOF)
- Neural Networks (e.g. Autoencoder)
- Isolation Forest

...

Strongly recommended library:
https://pyod.readthedocs.io/en/latest/



Anomaly Detection



https://www.mdpi.com/1424-8220/21/19/6679

Unsupervised Learning Tasks

- Clustering
- Dimensionality Reduction/Learning latent representations (Representation learning)
- Anomaly Detection
- Data Generation

Unsupervised Learning Tasks

- Clustering
- Dimensionality Reduction/Learning latent representations (Representation learning)
- Anomaly Detection
- Data Generation

Data Generation

- Generative Models

1. Variational Autoencoder
2. Generative Adversarial Network (Previous year lecture by N. Gentner)

- Generative Models aims at learning useful representations and to generate new samples from a complex distribution that they model where the data are sampled from

https://thispersondoesnotexist.com/

Data Generation

- Generative Models

1. Variational Autoencoder
2. Generative Adversarial Network (Previous year lecture by N. Gentner)

- Generative Models aims at learning useful representations and to generate new samples from a complex distribution that they model where the data are sampled from

https://thispersondoesnotexist.com/

Variational Autoencoder (VAE)

- In standard autoencoders, the latent space can be extremely irregular (close points in latent space can produce very different – often meaningless – patterns over visible units) so usually we cannot implement a generative process that simply samples a vector from the latent space and passes it through the decoder
- Possible fix: make the mapping probabilistic!

1. The encoder returns a distribution over the latent space instead of a single point
2. The loss function has an additional regularisation term in order to ensure a “better organization” of the latent space

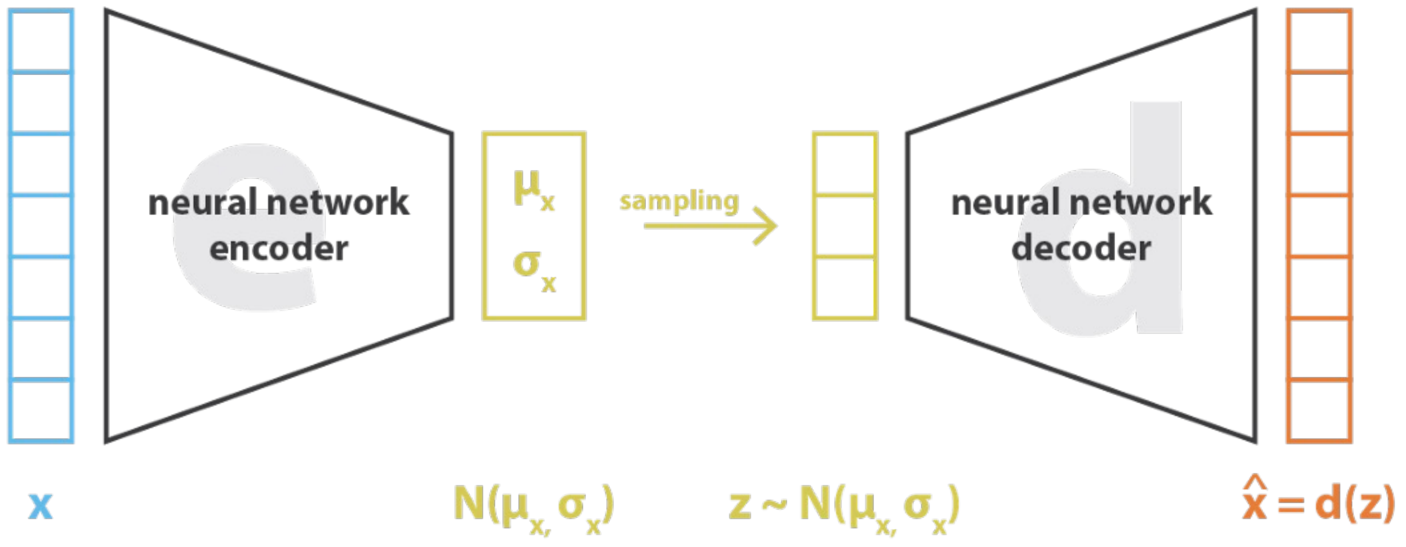
https://arxiv.org/abs/1312.6114

Variational Autoencoder (VAE)

- The encoded distribution is chosen to be a <u>multivariate Gaussian</u>, so that the encoder can be trained to estimate the means and covariance matrix
- This way we can regularize the loss function by forcing the latent distribution to be as close as possible to a standard Normal distribution

KL Divergence:

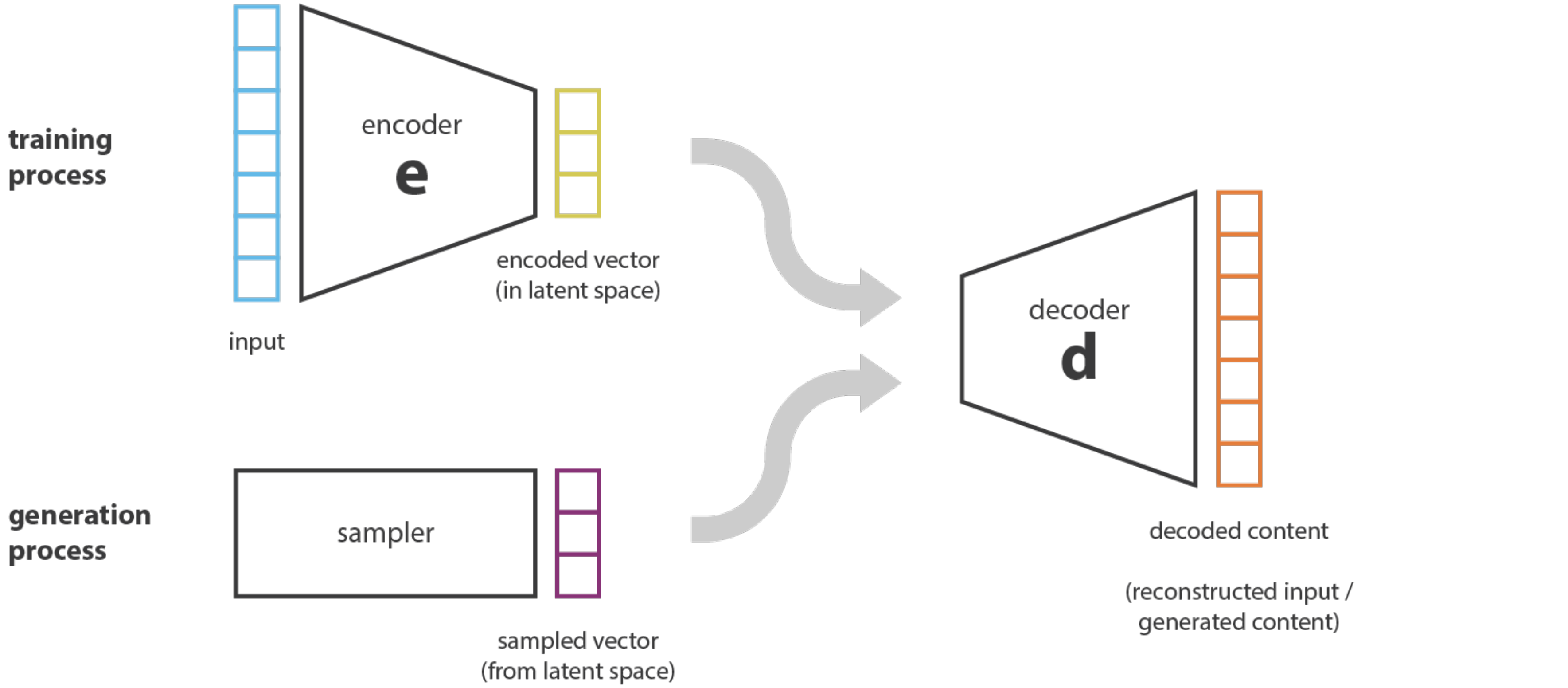
$$\boxed{D_{\mathrm{KL}}(P \parallel Q) = \int_{x_a}^{x_b} P(x) \log\left(\frac{P(x)}{Q(x)}\right) dx}$$



$$\text{loss} = \| x - \hat{x} \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I)\,]$$

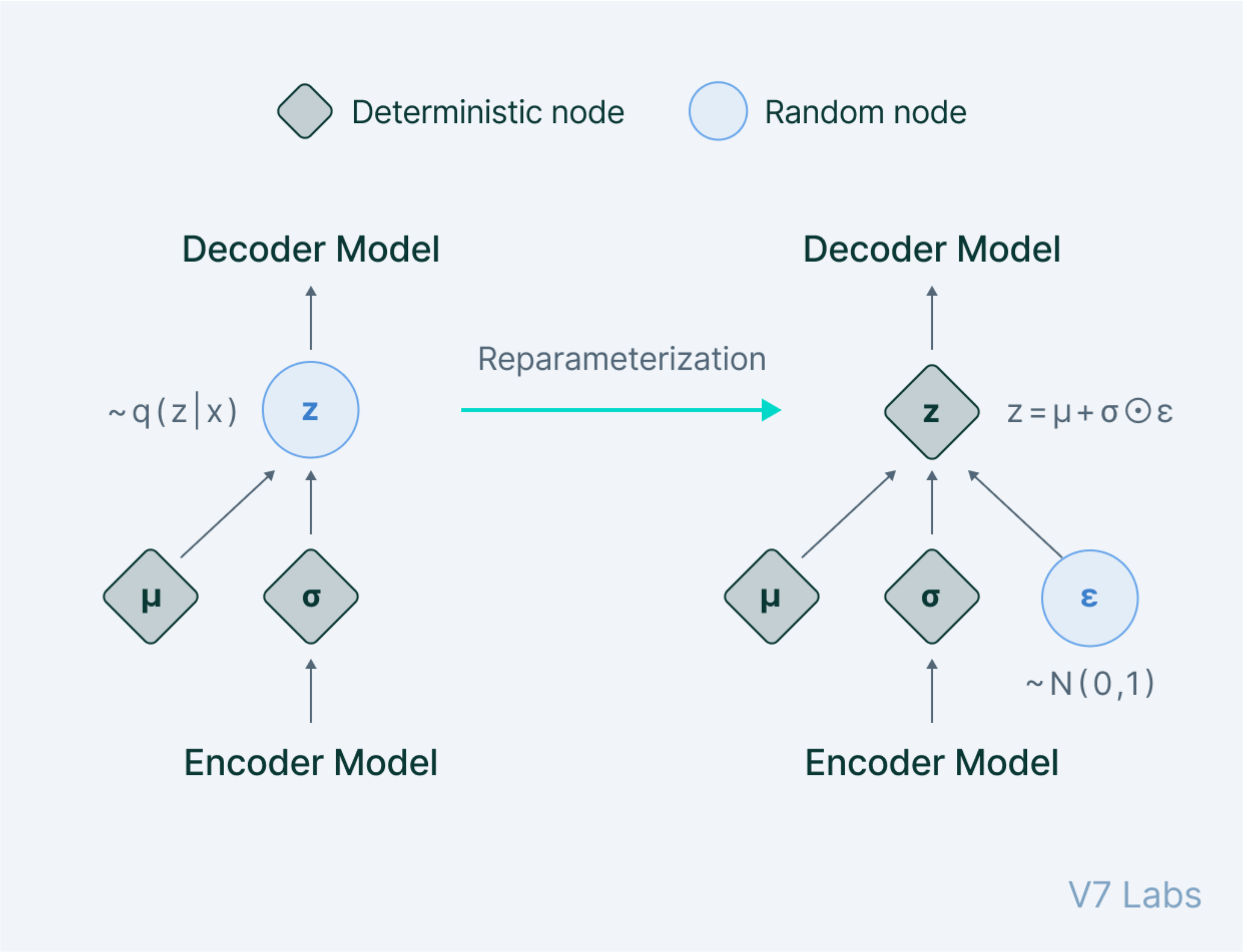
Variational Autoencoder (VAE)

- The encoded distribution is chosen to be a multivariate Gaussian, so that the encoder can be trained to estimate the means and covariance matrix
- This way we can regularize the loss function by forcing the latent distribution to be as close as possible to a standard Normal distribution



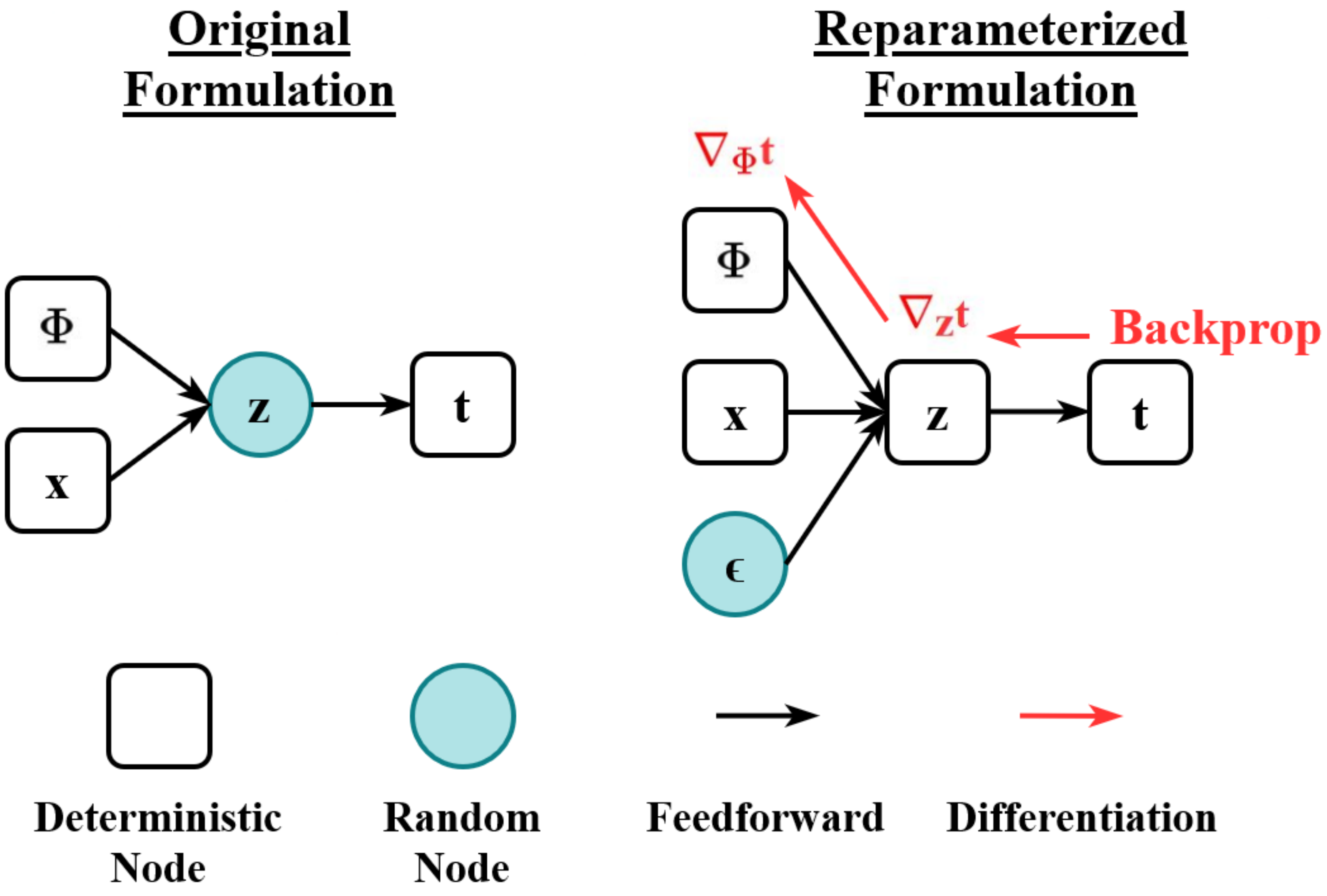
Reparametrization trick

- The latent representation is now defined by two vectors (means and covariance), so the encoder network has two (possibly partially overlapping) branches
- The covariance could just be a square matrix; however, to reduce computational complexity we assume that the multivariate Gaussian has a diagonal covariance matrix (i.e., latent variables are independent)
- Sampling is a discrete process, and we cannot use backpropagation! We need to re-parameterize z to make it differentiable



Reparametrization trick

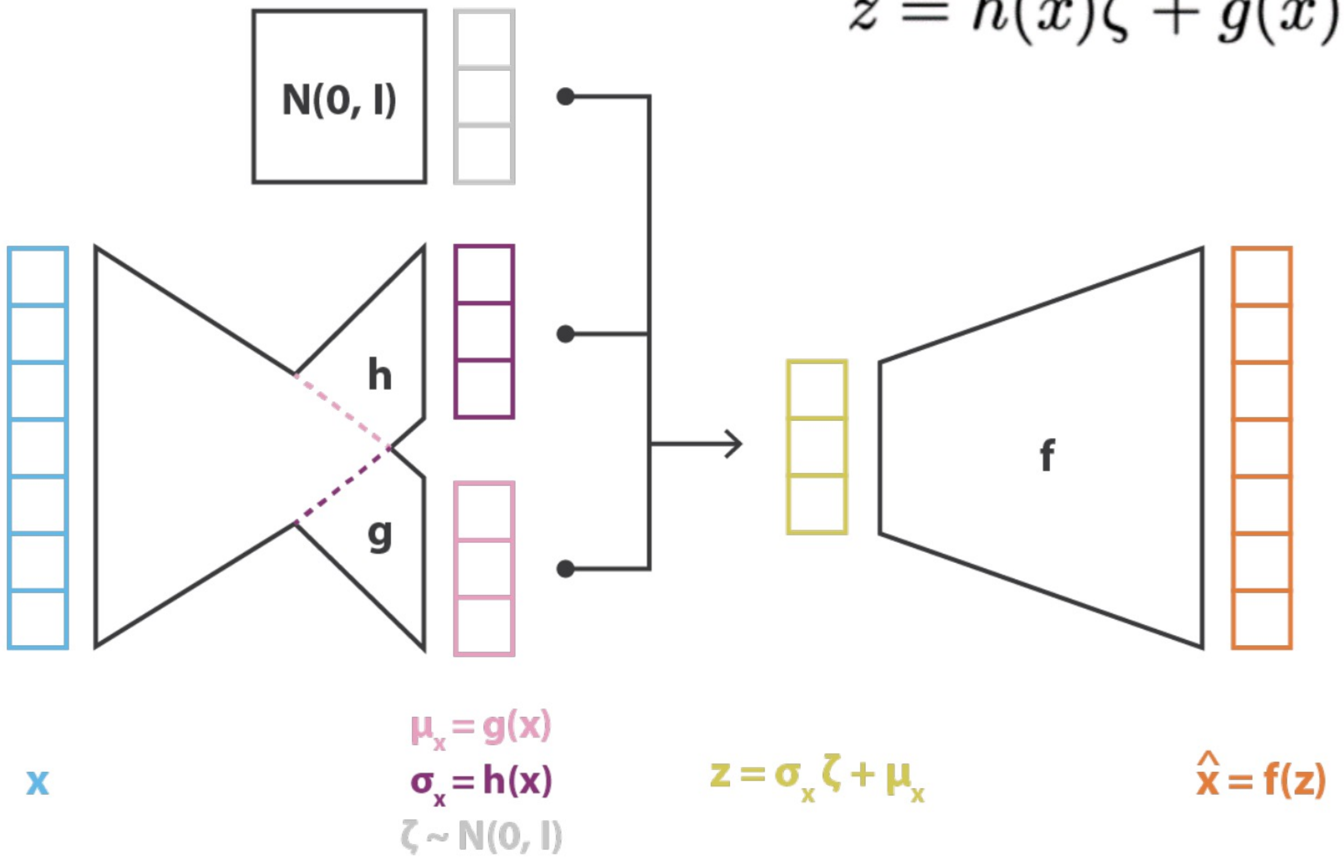
- The latent representation is now defined by two vectors (means and covariance), so the encoder network has two (possibly partially overlapping) branches
- The covariance could just be a square matrix; however, to reduce computational complexity we assume that the multivariate Gaussian has a diagonal covariance matrix (i.e., latent variables are independent)
- Sampling is a discrete process, and we cannot use backpropagation! We need to re-parameterize z to make it differentiable



A. Testolin 'Neural Networks and Deep Learning'

Reparametrization trick

$$z = h(x)\zeta + g(x) \qquad \zeta \sim \mathcal{N}(0, I)$$



$$\text{loss} = C\,\|\, x - \hat{x} \,\|^2 + \mathrm{KL}[\, N(\mu_x, \sigma_x), N(0, I) \,] = C\,\|\, x - f(z) \,\|^2 + \mathrm{KL}[\, N(g(x), h(x)), N(0, I) \,]$$

Variational Autoencoder (VAE)

- The regularization term indeed promotes the creation of a gradient over the latent representations, which allows to generate samples varying smoothly!

Disentangled VAE: $\boldsymbol{\beta}$-VAE

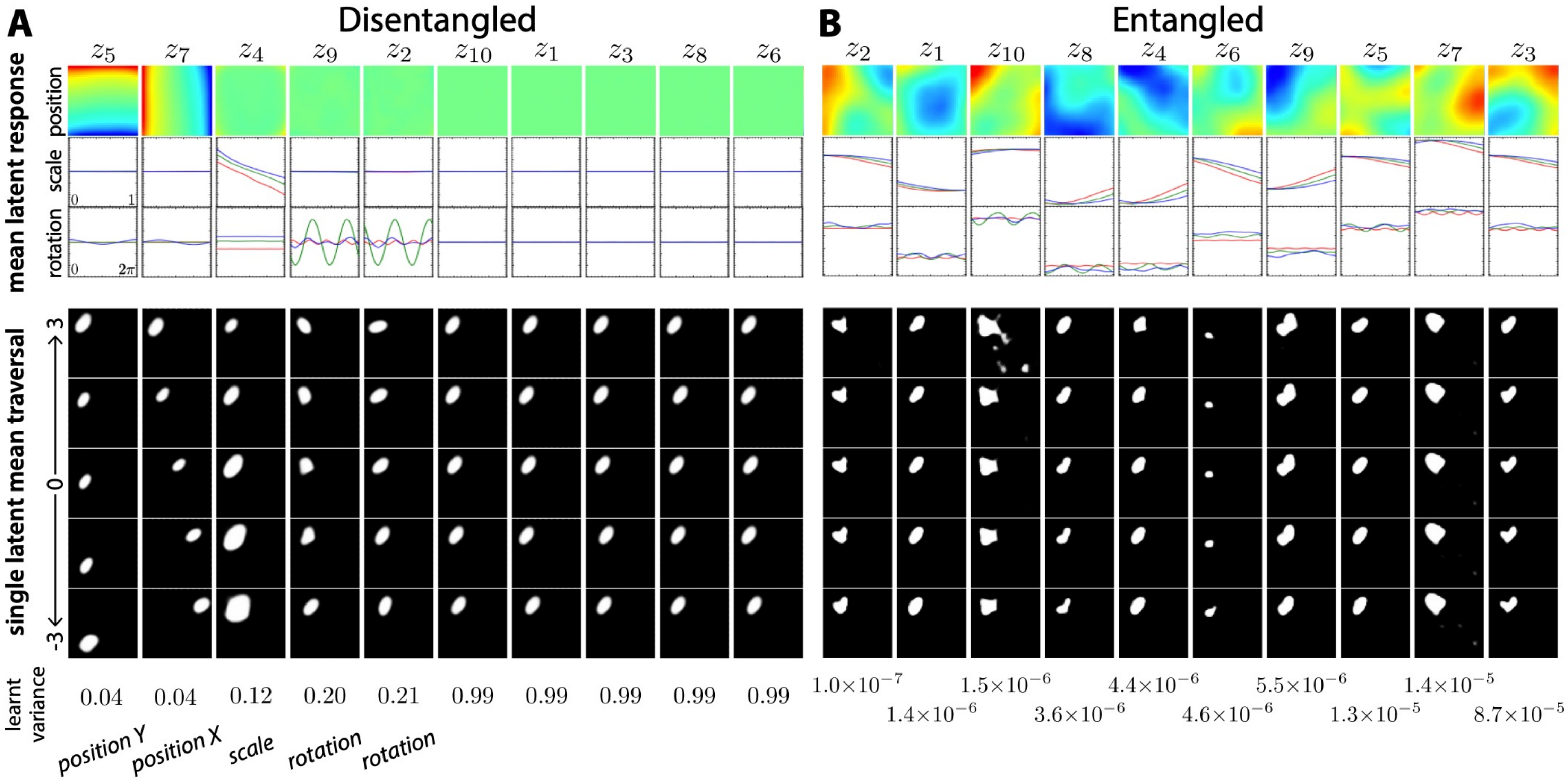
- VAE can be further extended to promote learning of more disentangled representations, which in some cases might encode independent latent factors of variation in the data distribution
- The final goal would be to have single latent units of z sensitive to changes in single generative factors (e.g., color of the hair) while being relatively invariant to changes in other factors (e.g., color of the skin)
- Basic idea: introduce a penalization term in the KL-divergence using a hyperparameter β > 1 that balances latent channel capacity and independence constraints with reconstruction accuracy (the higher the β, the more disentangled should be the representation)

$$\mathcal{L}(\theta, \phi, \mathbf{x}^{(i)}) = -\beta D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \,||\, p(\mathbf{z})\right) + \mathbb{E}_{q_\phi}(\mathbf{z}|\mathbf{x}^{(i)})\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

Disentangled VAE: β-VAE

https://arxiv.org/pdf/1606.05579.pdf

https://arxiv.org/pdf/1804.03599.pdf

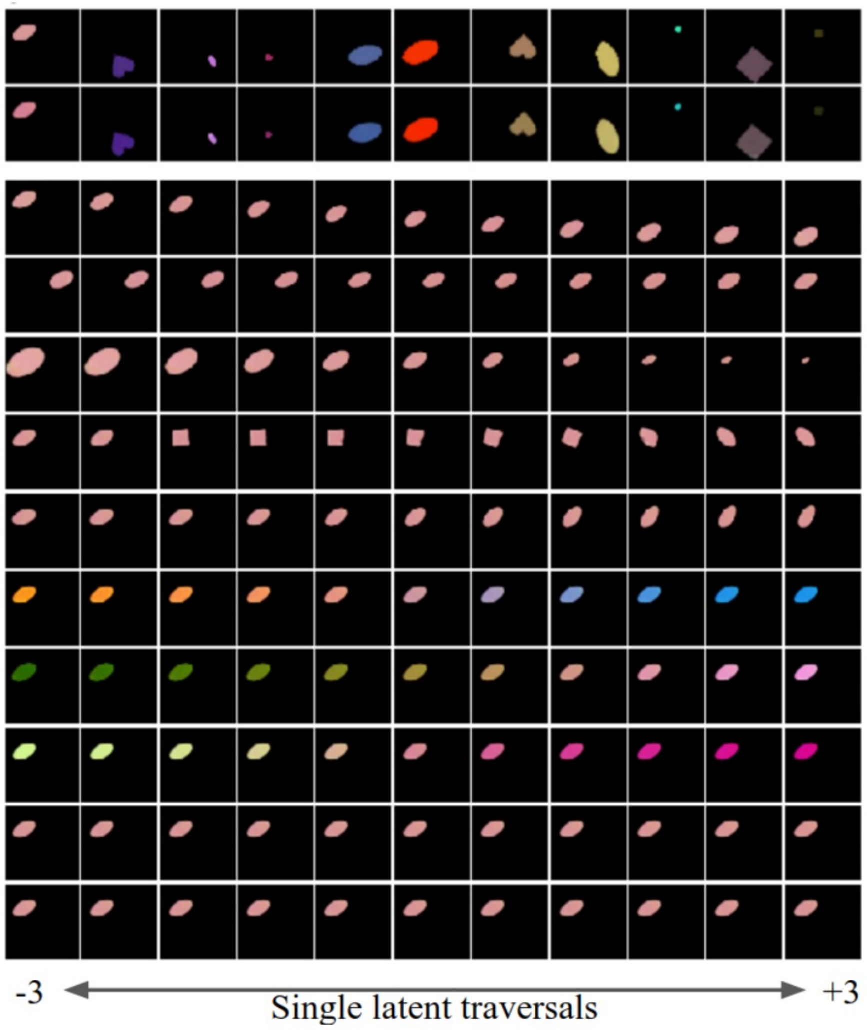


Disentangled VAE: β-VAE

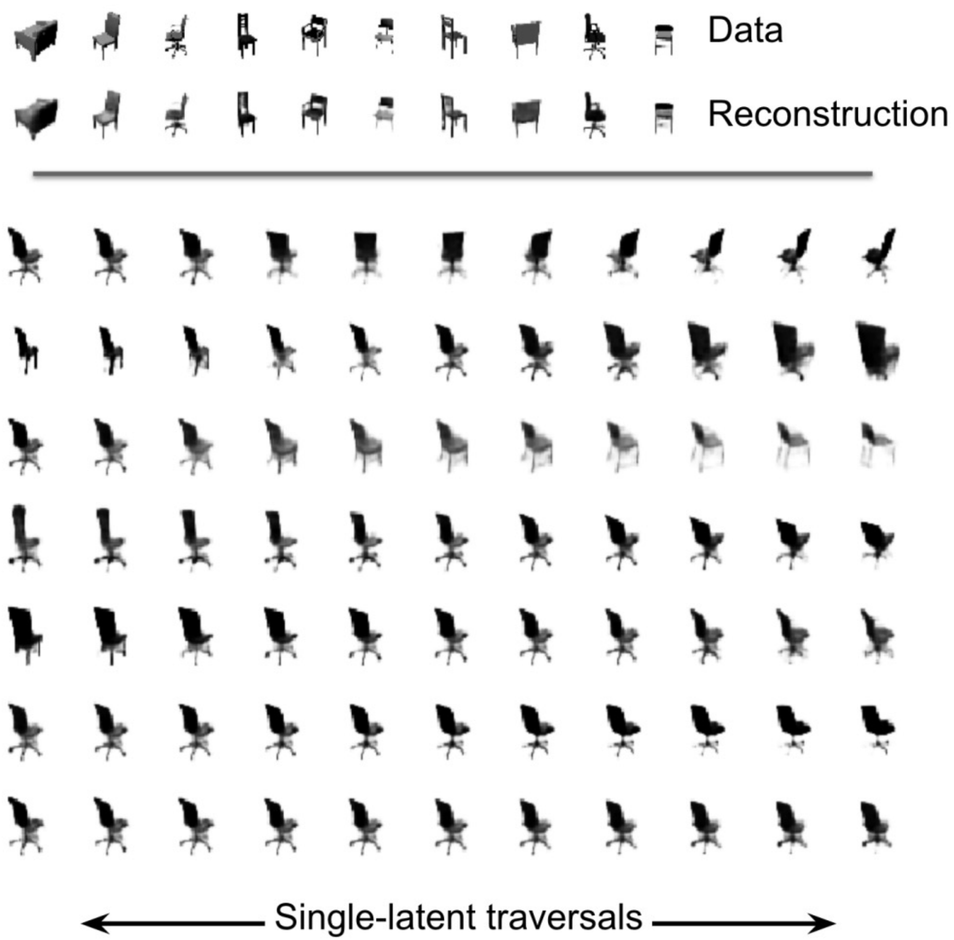
https://arxiv.org/pdf/1606.05579.pdf

https://arxiv.org/pdf/1804.03599.pdf

(a) Coloured dSprites



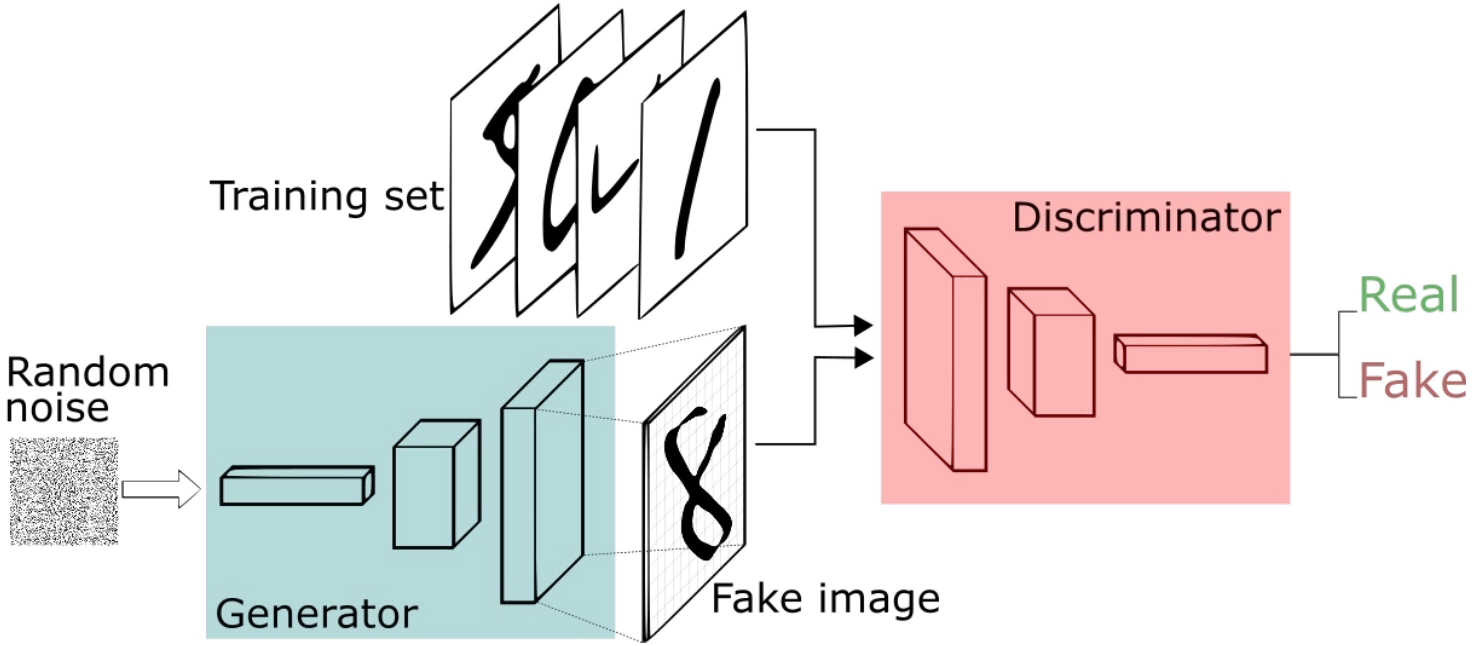
(b) 3D Chairs



Other generative approaches: GANs

- Generative Adversarial Networks

- You'll find a dedicated legacy lecture on the moodle page by N. Gentner



https://sthalles.github.io/intro-to-gans/

Thank you!

Gian Antonio Susto