Relevant Architectures (1)

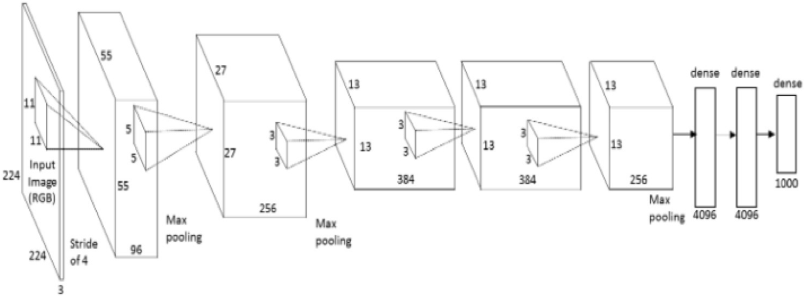
Relevant Architectures

- Encoder-classifier
  - Transfer Learning
- Encoder-decoder
- Autoencoder
- U-net
- Transformer (next lesson)
- Visual Transformer (next lesson)

Encoder-classifier

AlexNet

AlexNet Architecture (Krizhevsky, Sutskever e Hinton), winner of a NIPS contest in 2012.



Purely historical interest.

Pooling

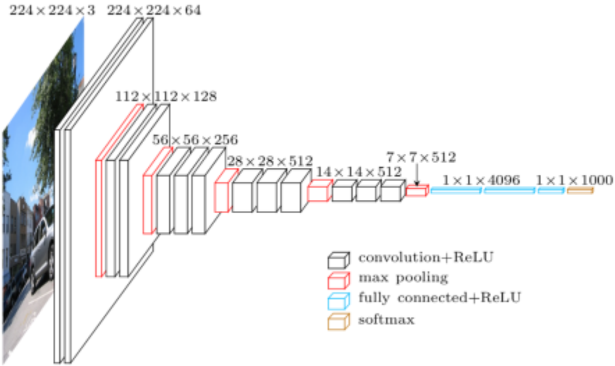
In deep convolutional networks, it is common practice to alternate convolutional layers with pooling layers, where each neuron simply takes the mean or maximal value in its receptive field.

This has a double advantage:

- it reduces the dimension of the output
- it gives some tolerance to translations

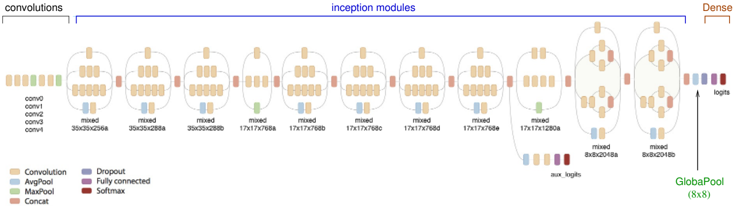
VGG

VGG 16 (Simonyan e Zisserman). 92.7 accuracy (top-5) in ImageNet (14 millions images, 1000 categories).



Picture by Davi Frossard: VGG in TensorFlow

Inception V3

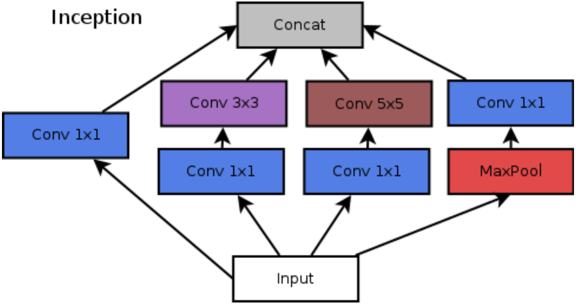


The convolutional part is a long composition of

inception modules

Inception modules

The networks is composed of inception modules (towers of nets):

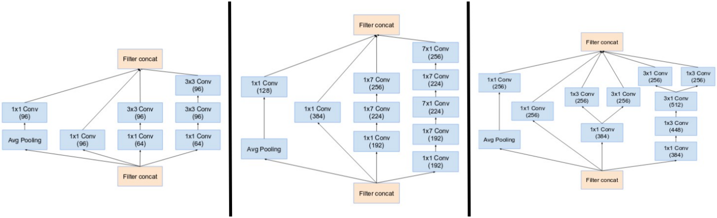


Video from the Udacity course "Deep Learning"

Variants

The point is to induce the net to learn different filters.

Many variants proposed and used over years:



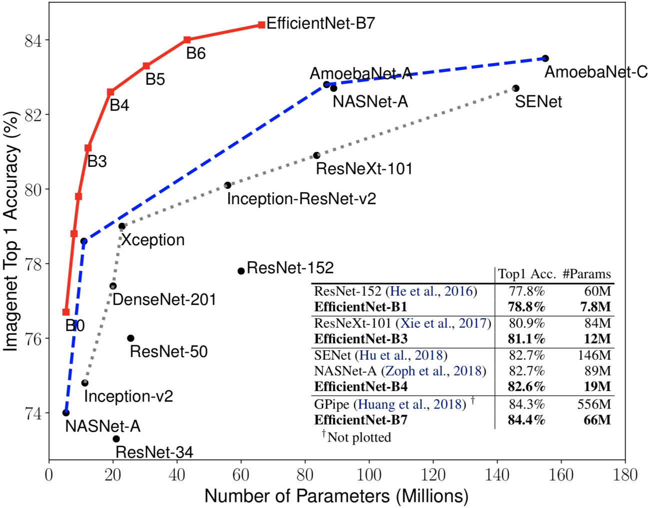
Xception, MobileNet, EfficientNet

These models focus on reducing the number of parameters for efficiency resasons (e.g. embedding on mobile devices)

- Xception
- MobileNet - a class of “light” models conceived to be deployed on mobile devices.
- EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.

One of the key tools are **Depth Separable Convolutions**.

Efficient Net

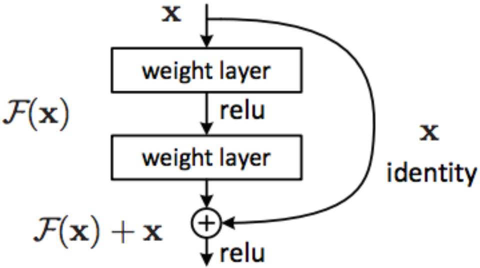


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018)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

† Not plotted

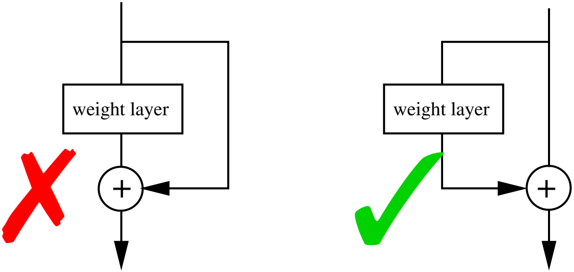
Residual Learning

Another recent topic is residual learning.

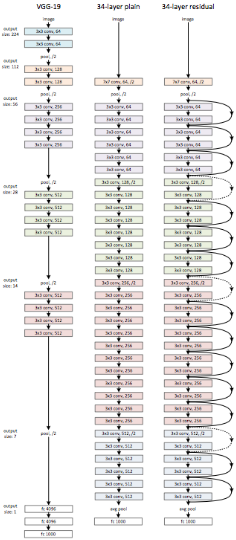


Instead of lerning a function $\mathcal{F}(x)$ you try to learn $\mathcal{F}(x) + x$.

The right intuition



Residual networks



you add a residual shortcut connection every 2-3 layers

Inception Resnet is an example of a such an architecture

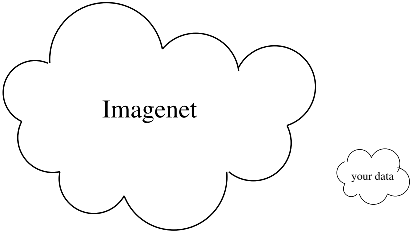
Why Residual Learning works?

- it seems to be a good idea to try to learn **non-linear corrections** over a linear baseline
- during back propagation, **the gradient at higher layers can easily pass to lower layers**, withouth being mediated by the weight layers, which may cause vanishing gradient or exploding gradient problem.

Demo: Cifar10 separation

[Demo]

A parenthesis:
Transfer Learning



Reusing Knowledge

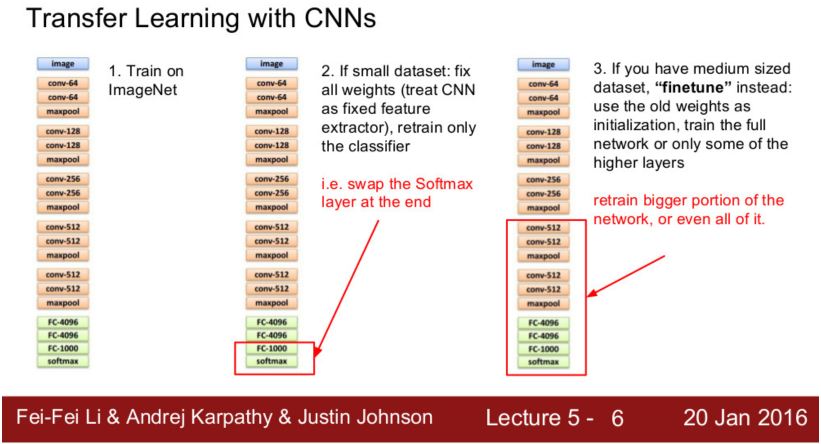
We learned that the first layers of convolutional networks for computer vision compute feature maps of the original image of growing complexity.

The filters that have been learned (in particular, the most primitive ones) are likely to be independent from the particular kind of images they have been trained on.

They have been trained on a huge amount of data and are probably very good.

It is a good idea to try to *reuse them* for other classification tasks.

Transfer Learning with CNNs



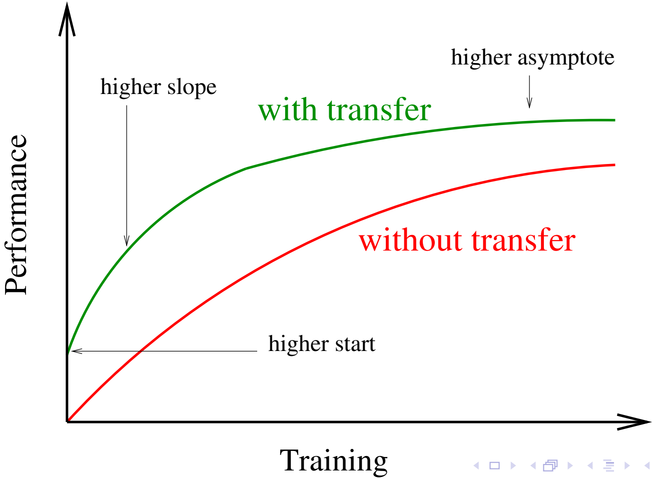
When Transfer Learning makes sense

transferring knowledge from problem A to problem B makes sense if

- the two problems have "similar" inputs
- we have much more training data for A than for B

What we may expect

Faster and more accurate training



Next argument

Encoder-Decoder

Encoder-Decoder

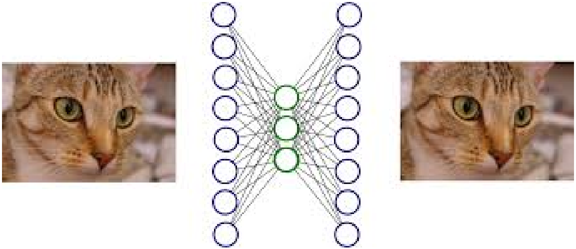
Encoder-Decoder models are a large class of models composed of two basic components:

Encoder transforming the input into a latent representation, typically a vector in some space, called the **latent space**.

Decoder reconstructing from the internal representation an explicit output into the visible space.

Autoencoders

An autoencoder is a net trained to reconstruct **input data** out of a learned internal representation



Usually, the internal representation has lower dimensionallity w.r.t. the input.

Demo: autoencoders in Keras

See Building autoencoders in Keras, on the Keras blog

Compression

Why is data compression possible, in general?

Because we exploit **regularities** (correlations) in the features describing input data.

If the input has a random structure (high **entropy**) no compression is possible

- random, lawless, uncompressible, high entropy
- ordered, lawfull, compressible, low entropy

Compression

Why is data compression possible, in general?

Because we exploit **regularities** (correlations) in the features describing input data.

If the input has a random structure (high **entropy**) no compression is possible

- random, lawless, uncompressible, high entropy
- ordered, lawfull, compressible, low entropy

Autoencoder's compression

When the internal layer has fewer units of the input, autoencoders implement as a form data compression, similar to not-linear PCA.

- **data-specific**: it only works well on data with strong correlations (e.g. digits, faces, etc.) This is different from traditional data compression algorithms
- **lossy**: the output is degraded with respect to the input. This is different from textual compression algorithms, such as gzip
- **directly trained** on unlabeled data samples. We usually talk of self-supervised training

Main applications of autoencoders

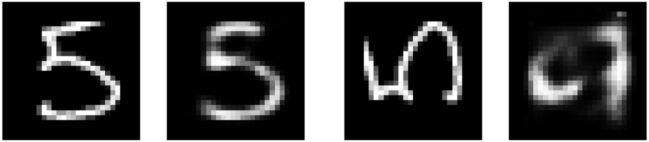
- data denoising
- anomaly detection
- feature extraction

Anomaly detection

Autoencoding is **data specific**: the autoencoder works well on data similar to those it was trained on.

If applied on different data (anomaly), it will perform poorly.

Example on mnist data with the autoencoder of the previous demo.
mean loss = 0.10, std: 0.03



loss = 0.14 OK!   loss = 0.27 LIAR!

U-net

Suggested reading:

U-Net: Convolutional Networks for Biomedical Image Segmentation. By O.Ronneberger, P.Fischer, T.Brox

U-net

The U-net is the de-facto standard for image-to-image processing.

Typical applications comprise

- segmentation
- denoising
- deblurring
- image restoration
- inpainting
- ...

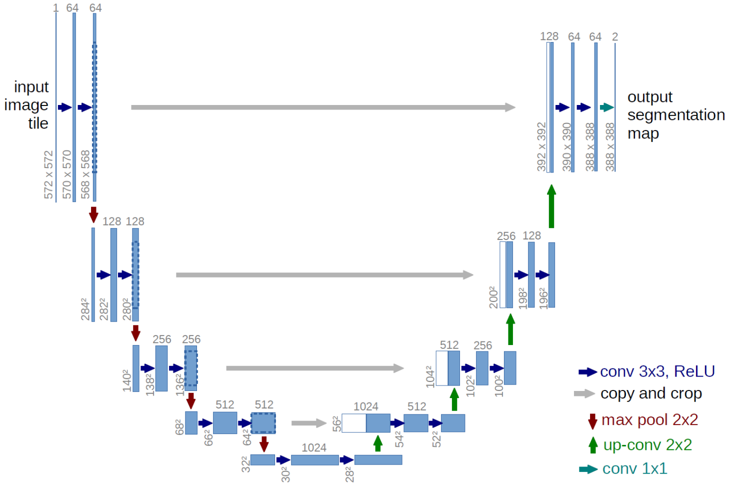
U-net architecture

The U-net complements an autoencoder structure with skip connections concatenating layers of the encoder to layers of the decoder at similar spatial resolution.

The auotencoder allows to obtain a holistic interpretation of the input (the what), while skip connections allows a more precise location and processing of features (the where).

The U-net allows to process together global features (extracted through the encoding process) with local features (merged with skip-connection).

U-net architecture



Demo: inpainting of satellite images

[Demo]