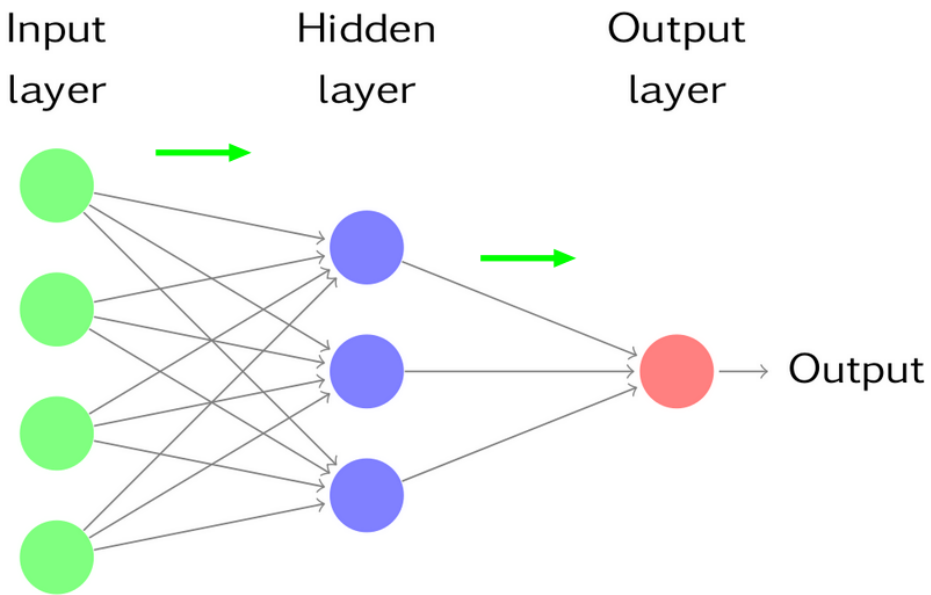
Convolutional neural networks

Victor Kitov

v.v.kitov@yandex.ru

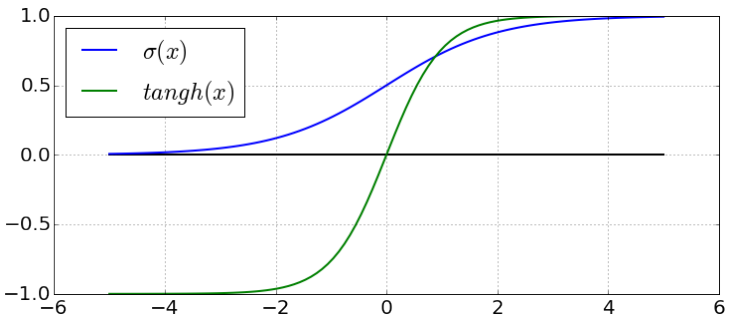
Mutlilayer perceptron



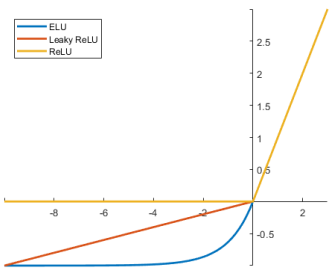
- Universal approximator (at least 2 inner layers with sufficient #neurons, non-linear activations)
- Fully-connected set of connections make MLP applicable only for low-dimensional input.

Activations

- sigmoid, tangh activations have gradient ≈ 0 in most cases.



- ReLU, LeakyReLU, ELU mostly don't have problems with gradient ≈ 0.



Final layer output

- Regression: output input I.
- Classification:

$$p(y=+1|x)=\frac{1}{1+e^{-I}} \qquad y\in\{+1,-1\}$$

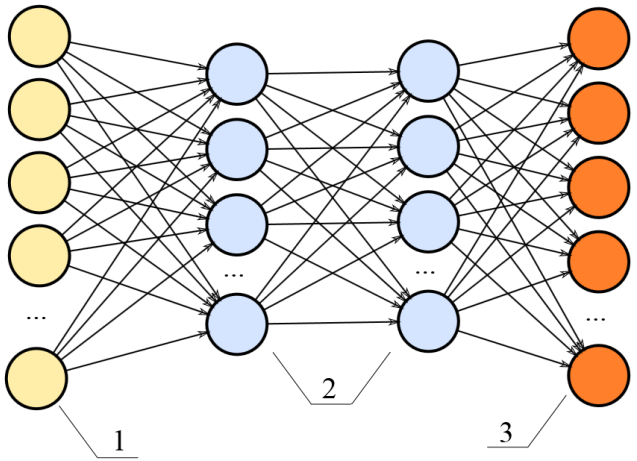
$$p(y=k)=SoftMax_k\left(I_1,...I_C\right)=\frac{e^{I_k}}{\sum_{c=1}^{C}e^{I_c}},\ y\in 1,2,...C$$

  - may use hinge loss if need only class label predictions
  - for probabilistic outputs - fit using maximum likelihood:

$$\prod_{n=1}^{N}p(y_n|x_n)\to\max_w \quad\Longleftrightarrow\quad \sum_{n=1}^{N}\ln p(y_n|x_n)\to\max_w$$

$$-\sum_{n=1}^{N}\sum_{c=1}^{C}\ln p(y=c|x_n)\mathbb{I}[y_n=c]\to\min_w \quad \text{(cross entropy loss)}$$

Autoencoder



$$\sum_{n=1}^{N} \|\widehat{y}(x_n) - x_n\|^2 \to \min_w$$

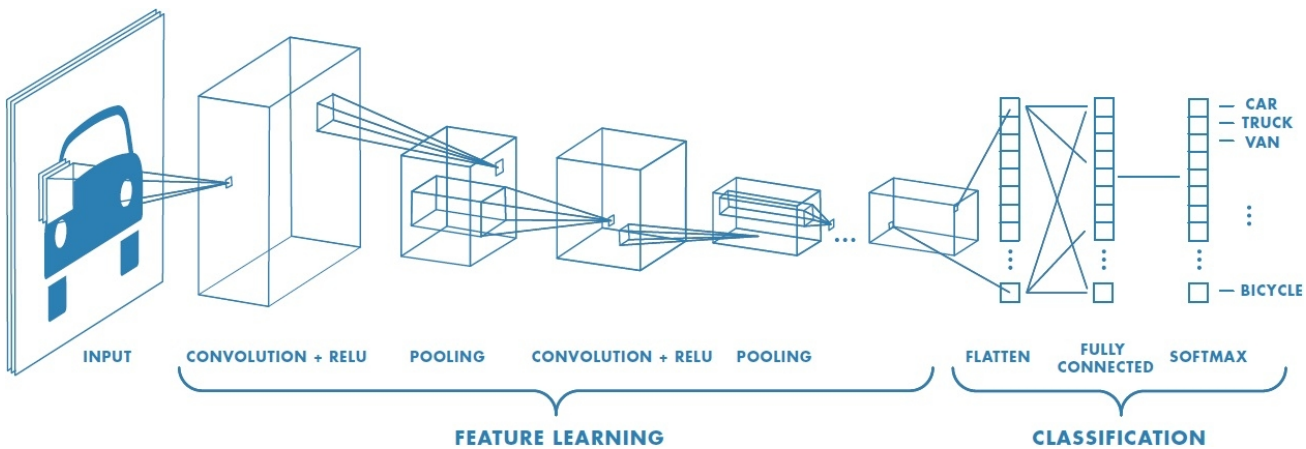
- Unsupervised learning.
- Non-linear dimensionality reduction.
- May initialize supervised model with encoder part of autoencoder.

Network training

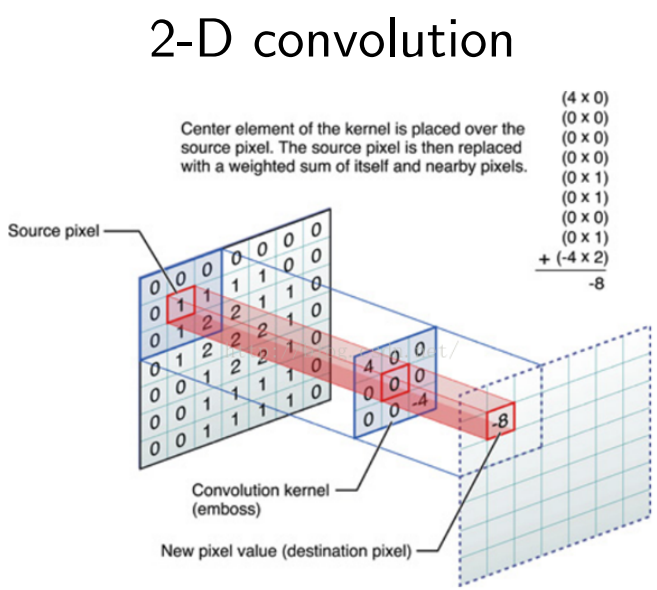
- Try Adam with different learning rates.
- Decrease learning rate after reaching plateau region.
- To combat overfitting use:
  - simplified architecture
  - L_2 regularization
  - early stopping
  - DropOut
  - Batch normalization (batch size should be sufficient)
  - pretrain model on related task with larger training set
    - e.g. ImageNet for images

Convolutional neural networks

- Convolutional neural network (classification):
  - Used for image analysis
  - Uses convolutional layers+elementwise non-linearitiy+pooling layers. MLP at the end.



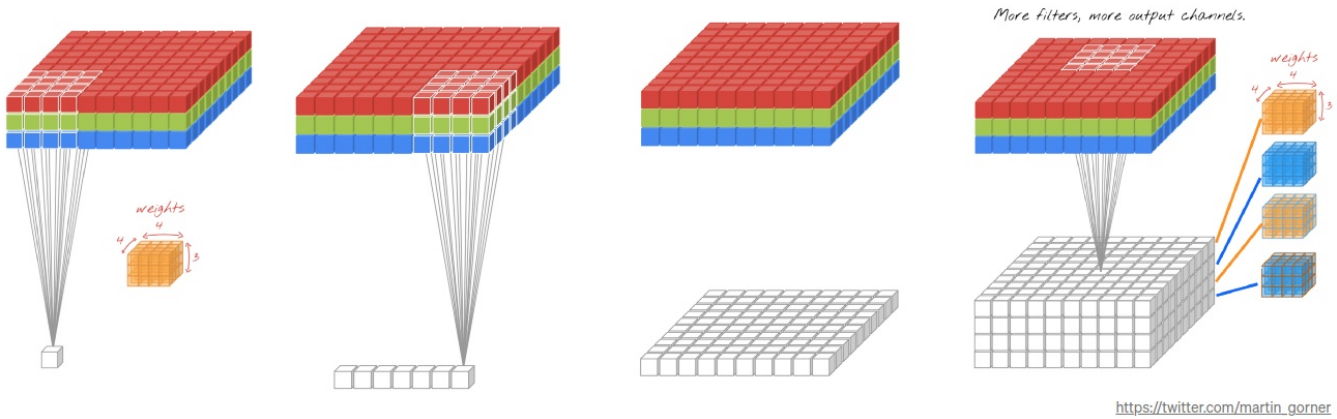
2D convolution



- Extracts local linear feature.
- Stacked convolutions+non-linearity transformations extract more complex features.
- Uses connections only to close neurons+weight sharing.

3D convolution

3D convolution



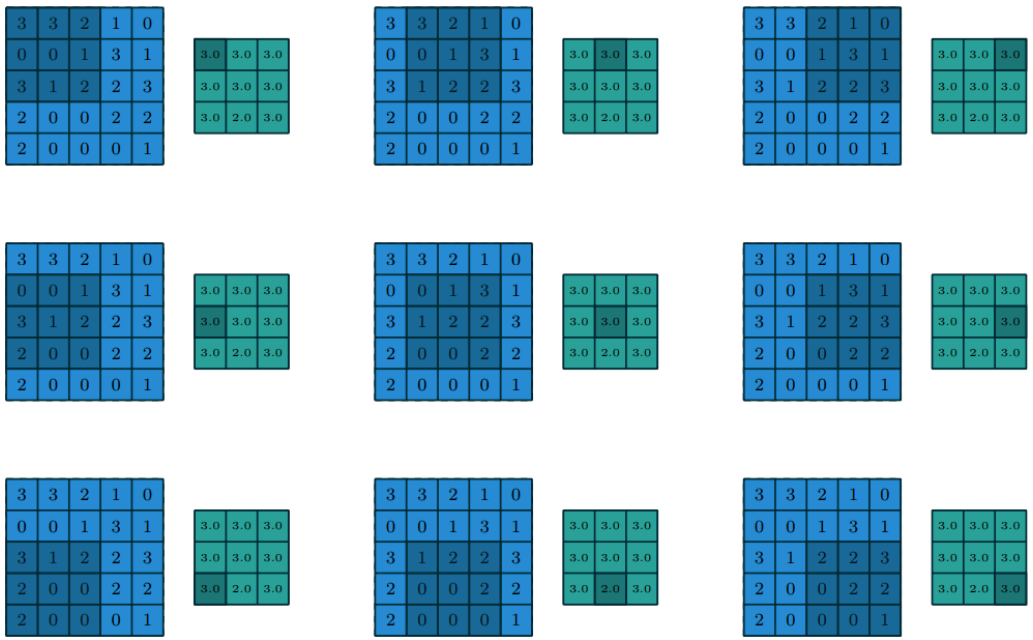
$$out3D(x,y,c) =$$

$$\sum_{i=-n}^{n}\sum_{j=-n}^{n}\sum_{c=1}^{C} K(i+n+1, j+n+1, c)in(x+i, y+j, c) + b,$$

$$K \in \mathbb{R}^{(2n+1)x(2n+1)}, \; b \in \mathbb{R}$$

Using different convolutions may control #output layers.

Pooling



- max pooling: feature is somewhere in the region
- avg. pooling: average feature presence in the region
- pooling implements invariance to small transitions on the image.

Change of spatial size

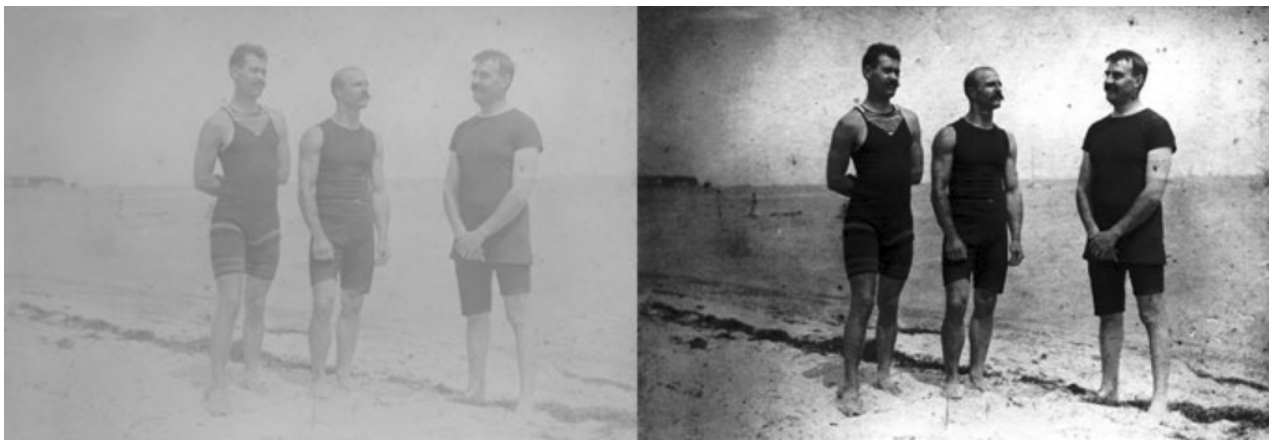
- Downscaling: strided pooling or strided convolution.
- Upscaling: apply convolution to enlarged input
  - transposed convolution (padding input values, "bed of nails")

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & b & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & c & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

  - simple scaling (nearest neighbours or rescaling with smoothing)

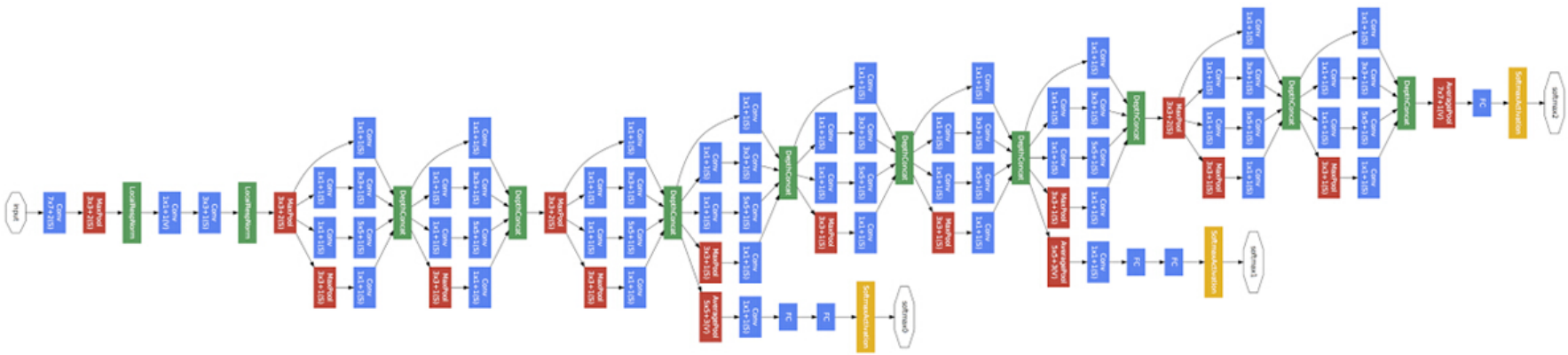
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \longrightarrow \begin{pmatrix} a & a & b & b \\ a & a & b & b \\ c & c & d & d \\ c & c & d & d \end{pmatrix}$$

Use dataset augmentation



- Scaling&cropping, rotations, translations, brightness, contrast, add noise, small color changes.

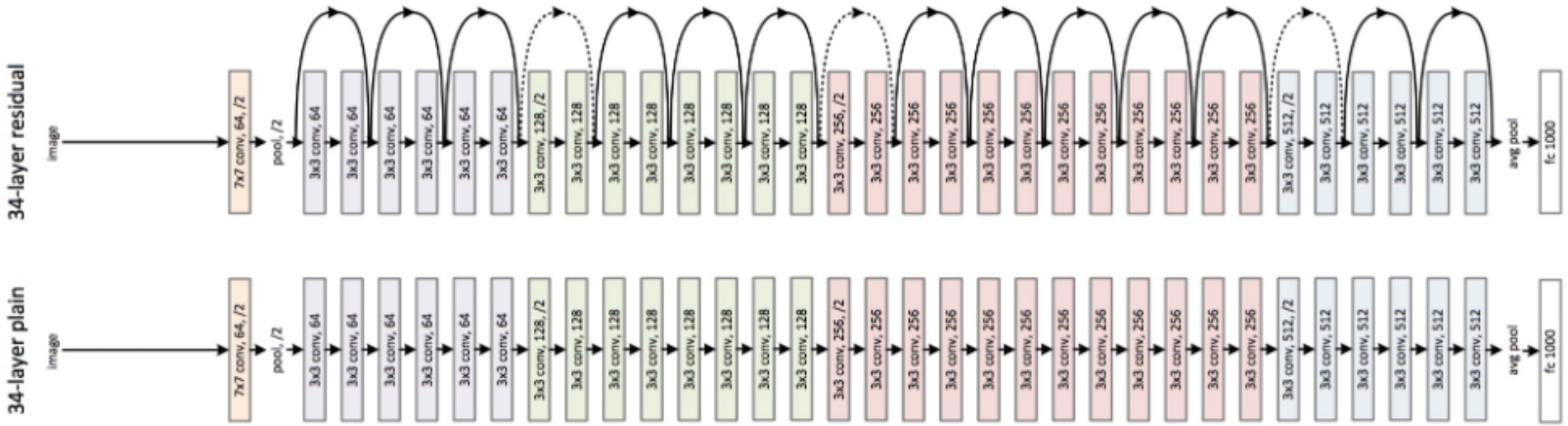
GoogleNet



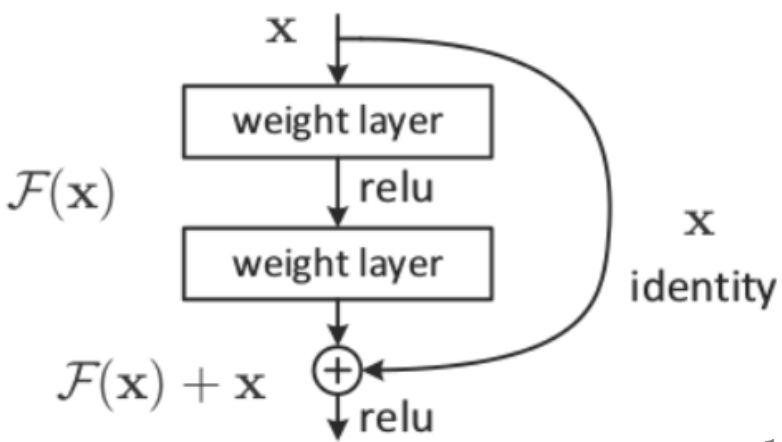
- add intermediate outputs during training
- reduce computation and # parameters by 1x1 convolutions

ResNet

ResNet vs. plain network



ResNet building block:

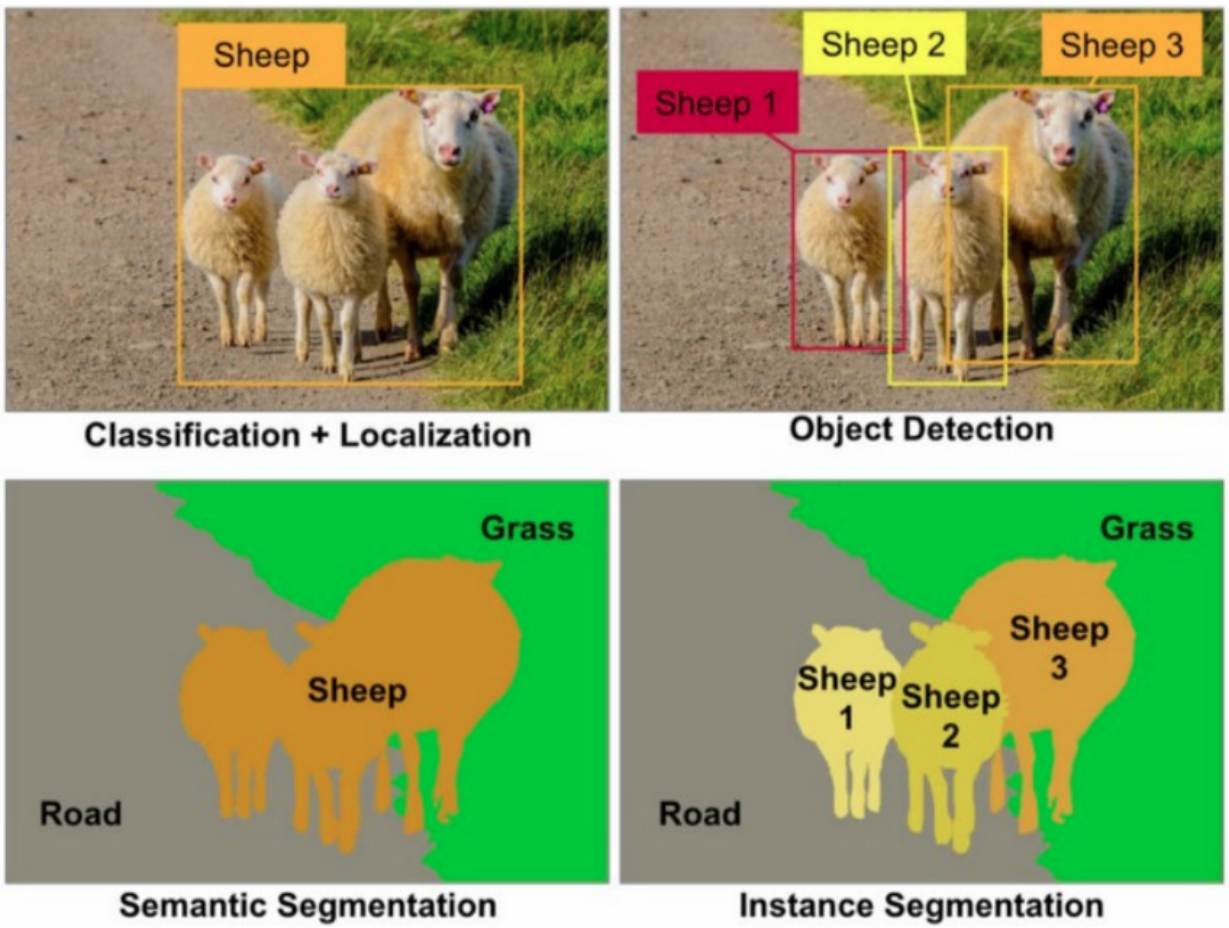


Identity connection allows to:

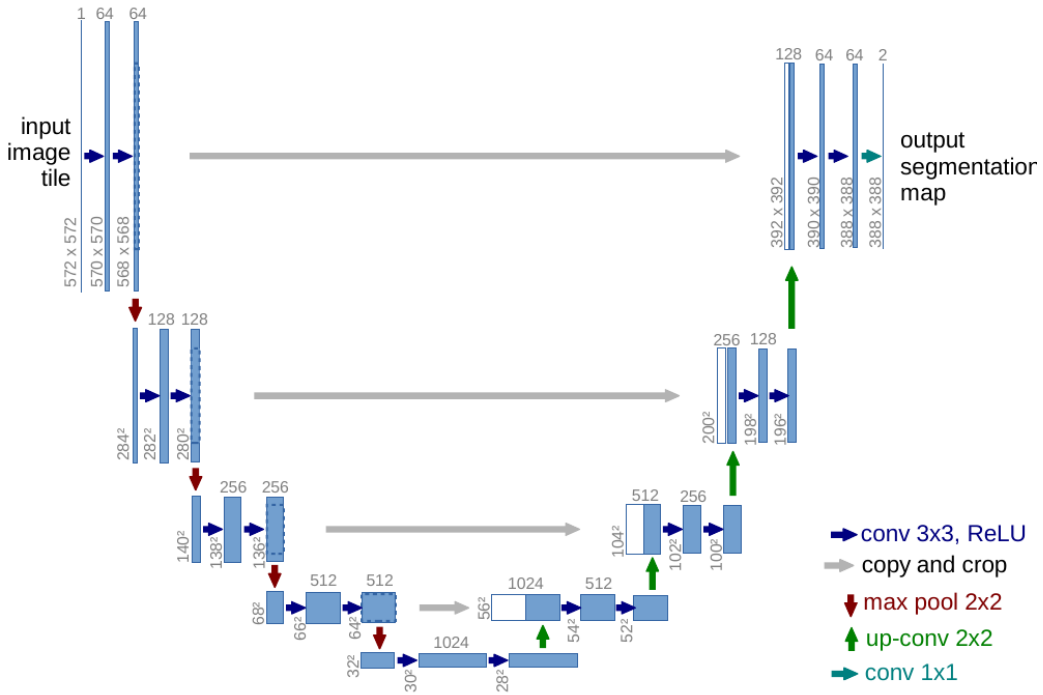
- better propagate gradient backwards
- init non-linear path with small weights

Major image tasks except classification

Different image tasks



Semantic segmentation



- Fully convolutional, softmax at every output pixel.
- Unite
  - coarse high level feature map (object class)
  - detailed low level feature map (to reconstruct boundaries)

Other topics

- Style transfer
- Object detection
- Generative adversarial networks
- Word embeddings
- Recurrent neural networks