EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Machine Learning in Graphics and Vision

- Convolutional Neural Networks -

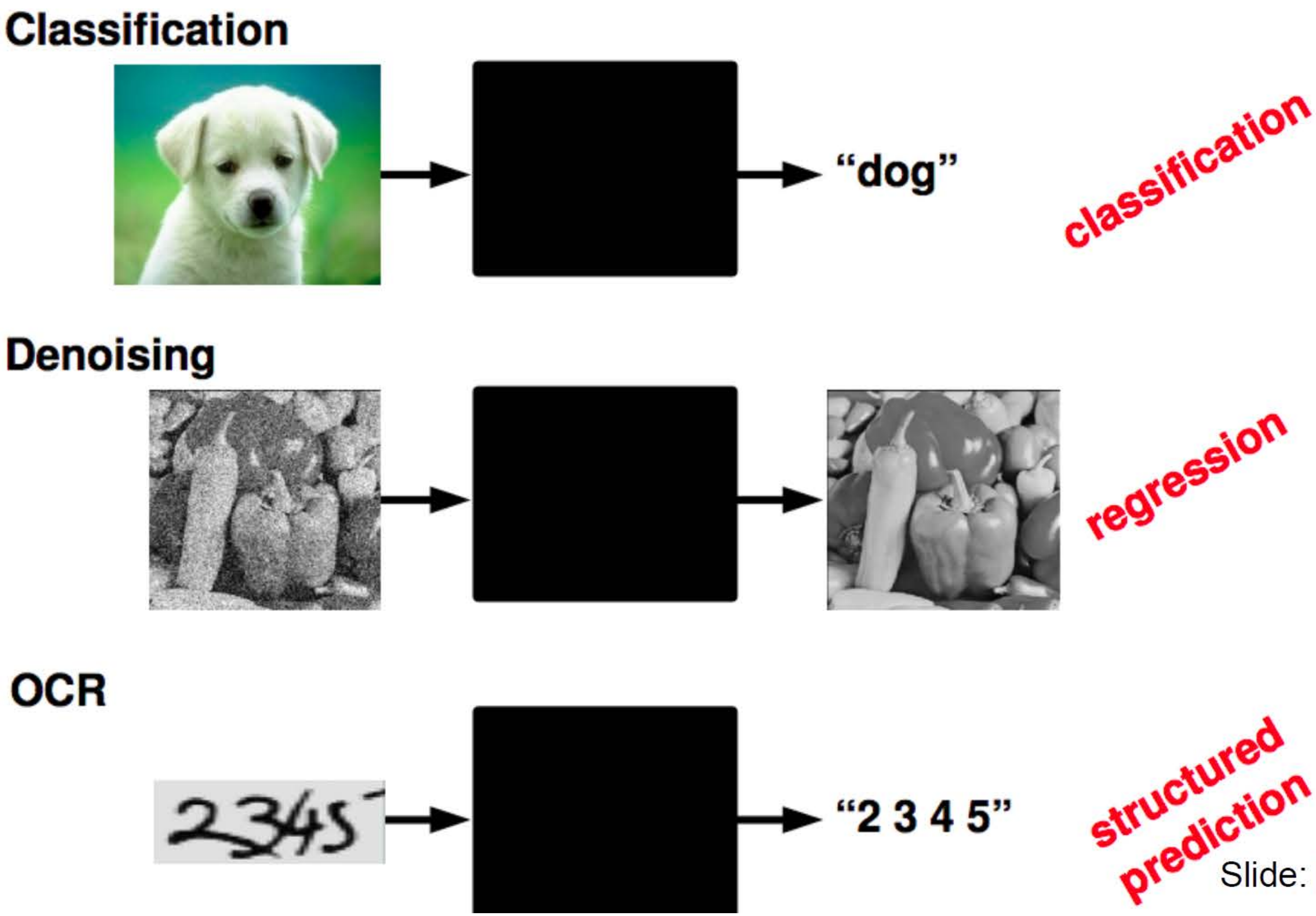
SoSe 2018

Hendrik Lensch

Overview

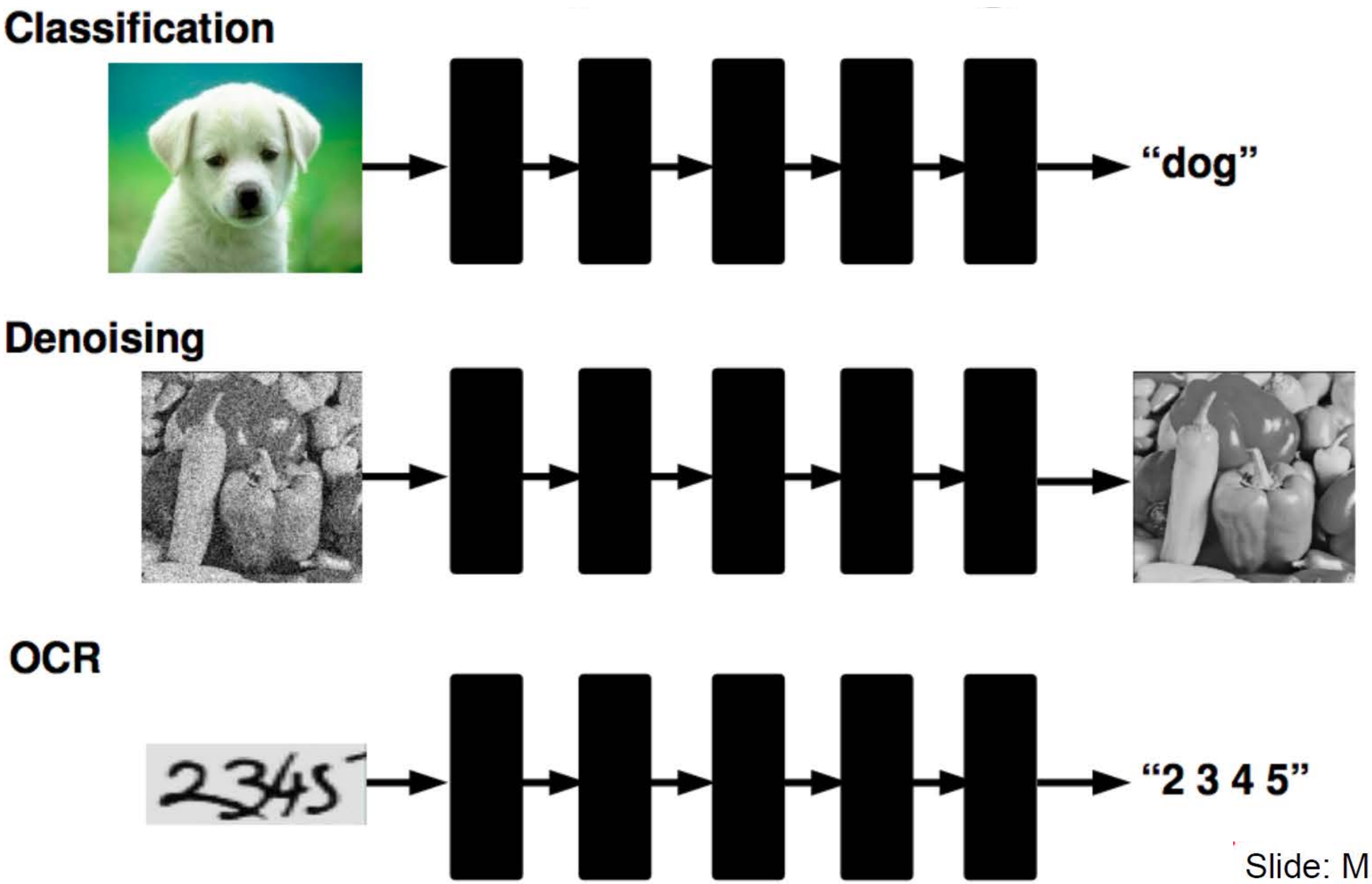
- ConvNets
- Dilated convolutions
- Dropout
- Batchnorm
- VGG19
- Resnet
- Autoencoder
- U-Net, Skip-Connection
- Image classification
- Semantic segmentation
- Denoising

Supervised Learning: Examples



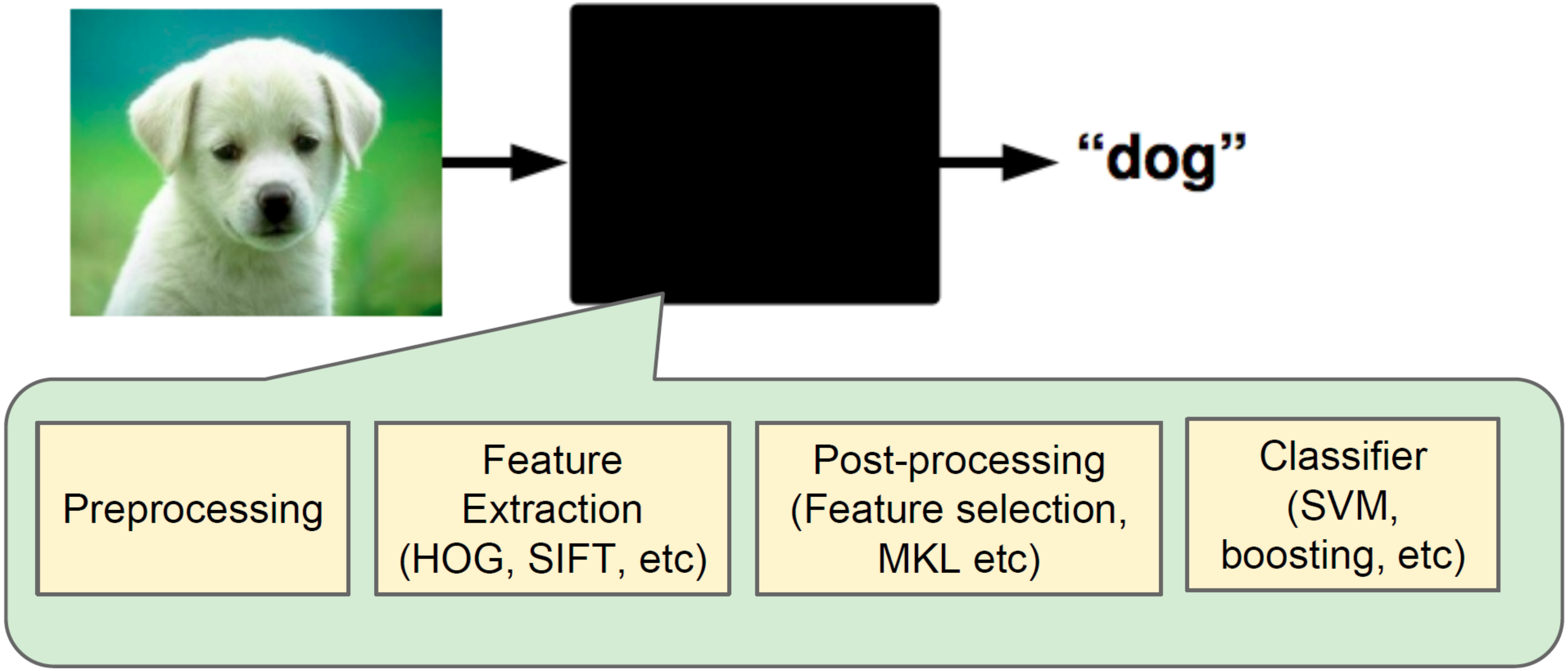
Slide: M. Ranzato

Supervised Deep Learning

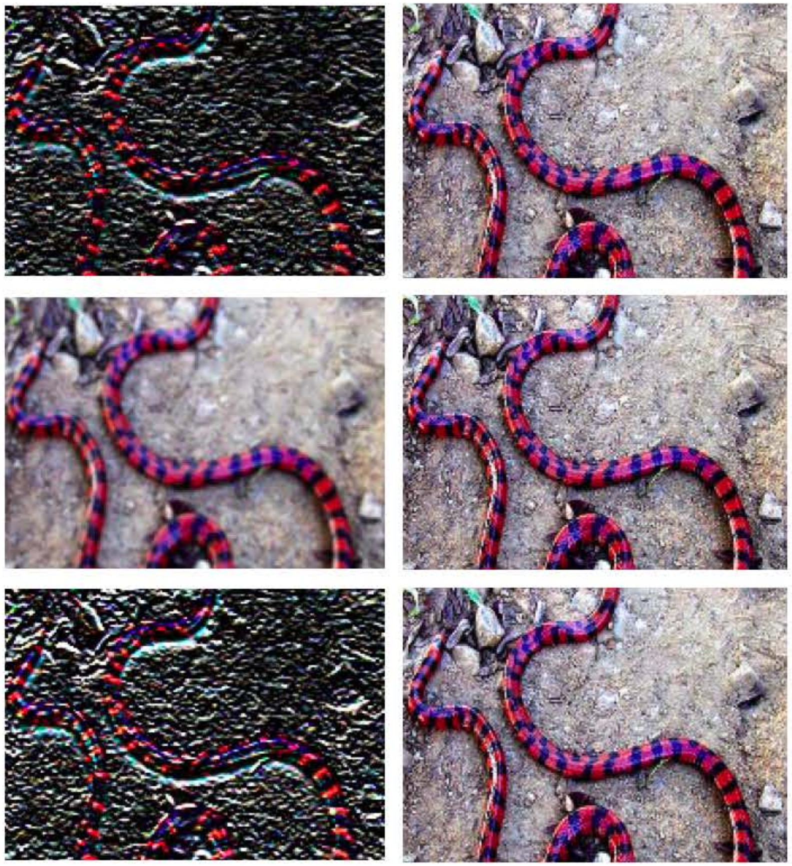


Slide: M. Ranzato

Traditional, Tailored Recognition approach



2D Convolution



Prewitt

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

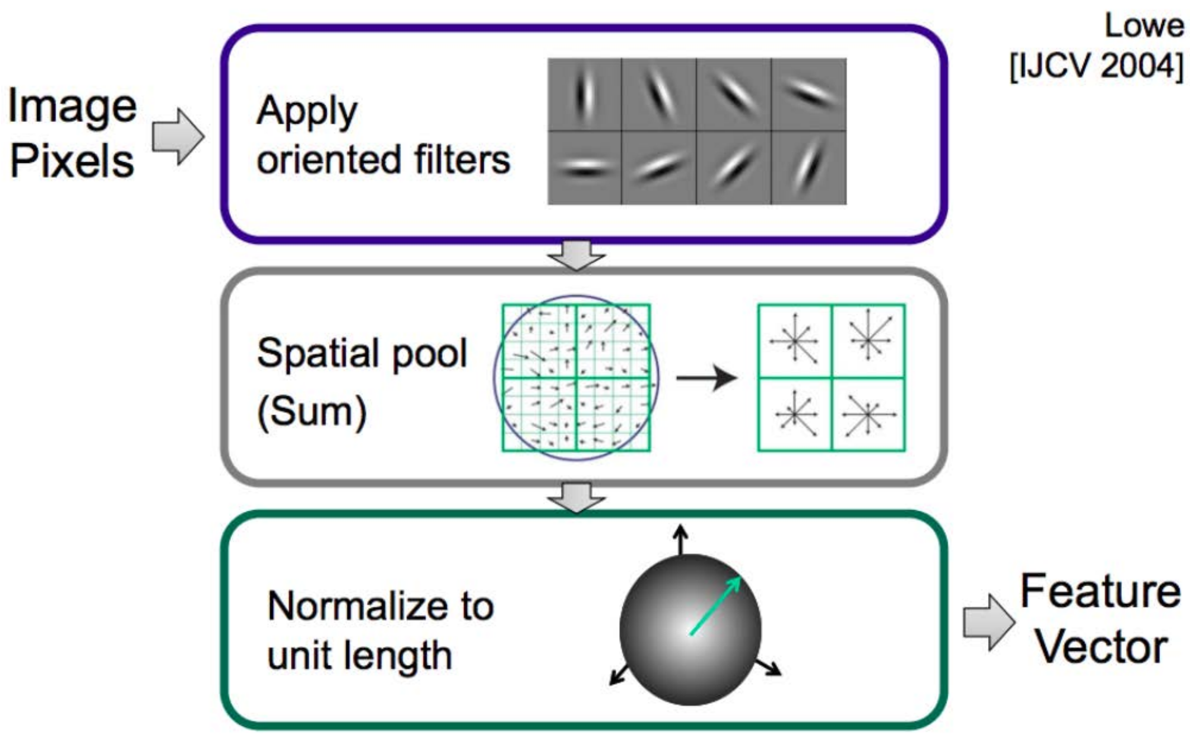
Blur

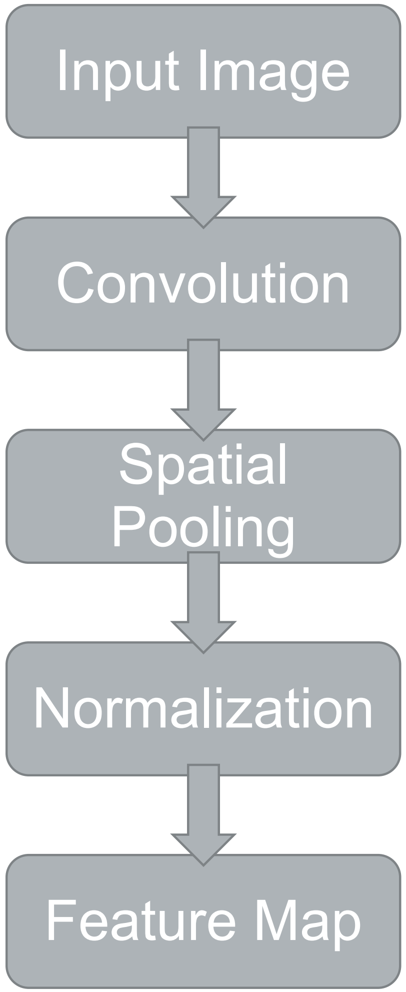
$$\frac{1}{N}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel

$$\frac{1}{N}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Compare to SIFT





- Missing scale and orientation invariance

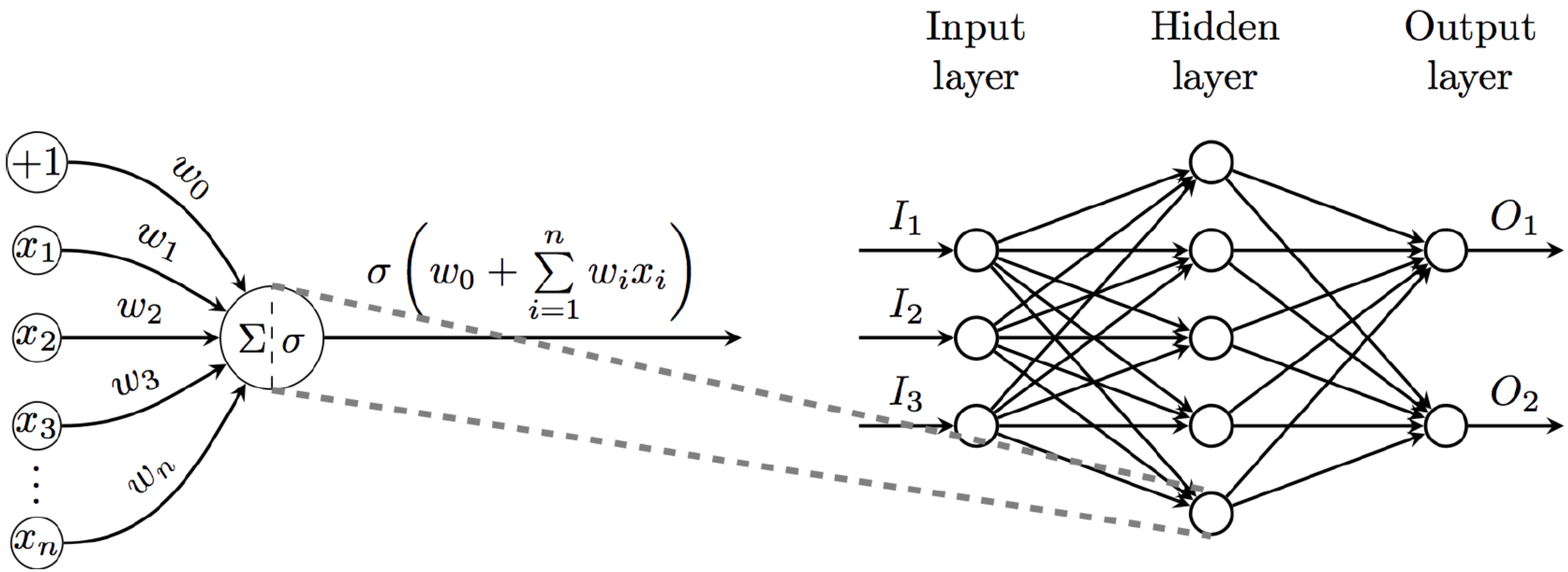
Hand-Crafted Features and Classification

- Most critical for accuracy
- Time-consuming development
- Best feature for the task
- Developing better features
- Features and Classfication should interoperate

- **Let‘s learn the feature representation directly from data**

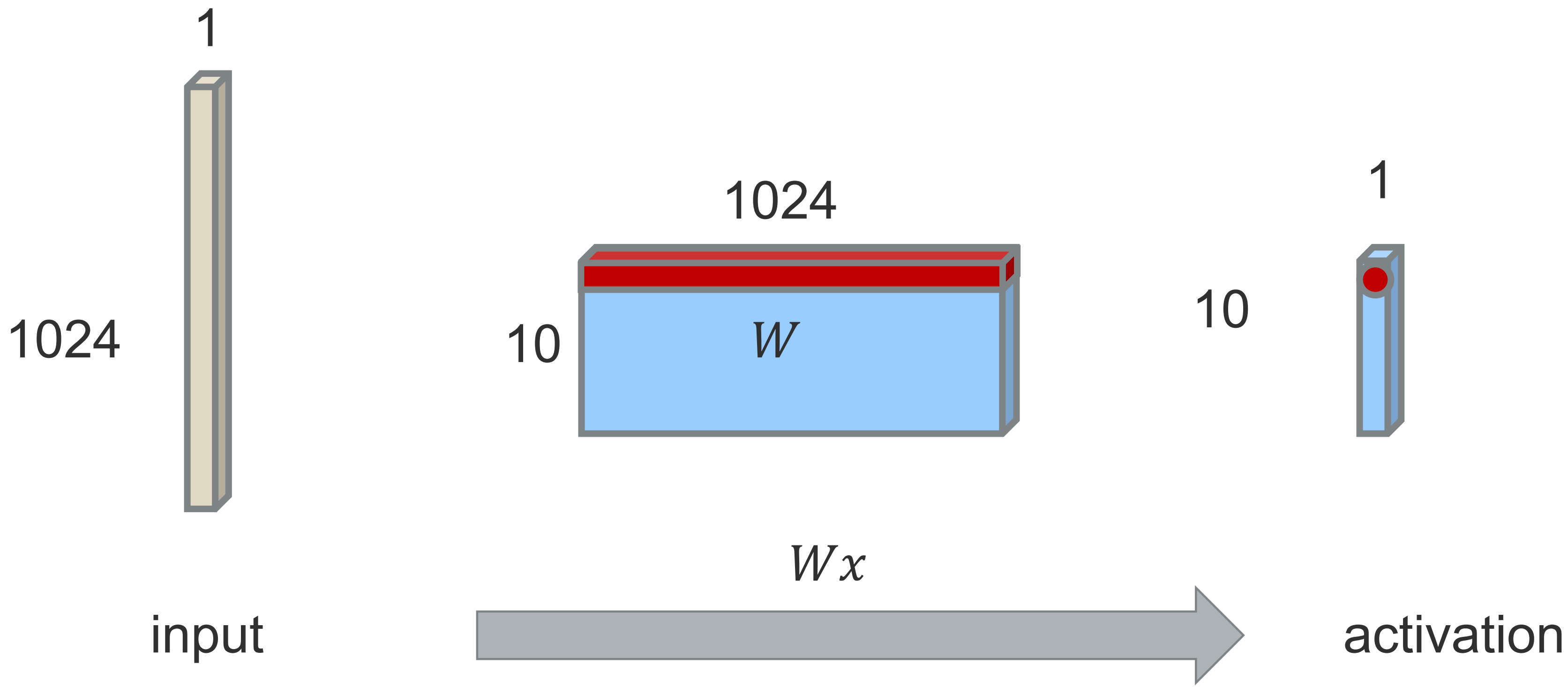
Recap: Multilayer Perceptron

- 3 layers
- Linear mapping (dot product + bias) $w_0 + \sum_{i=0}^{n} w_i x_i$
- Activation function, e.g. sigmoid $\sigma(x) = \frac{1}{1+\exp(-x)}$



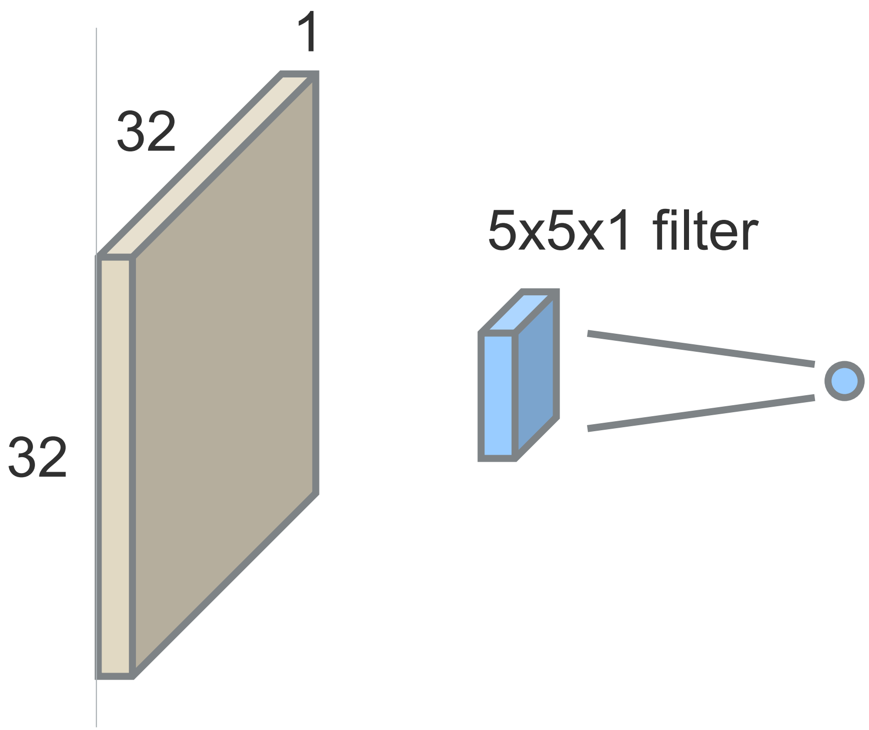
Perceptron – Fully Connected

- From 1024 to 10 neurons
- Each connection represented by its own weight
- Corresponds to matrix vector multiplication (dot product for each ouput element)



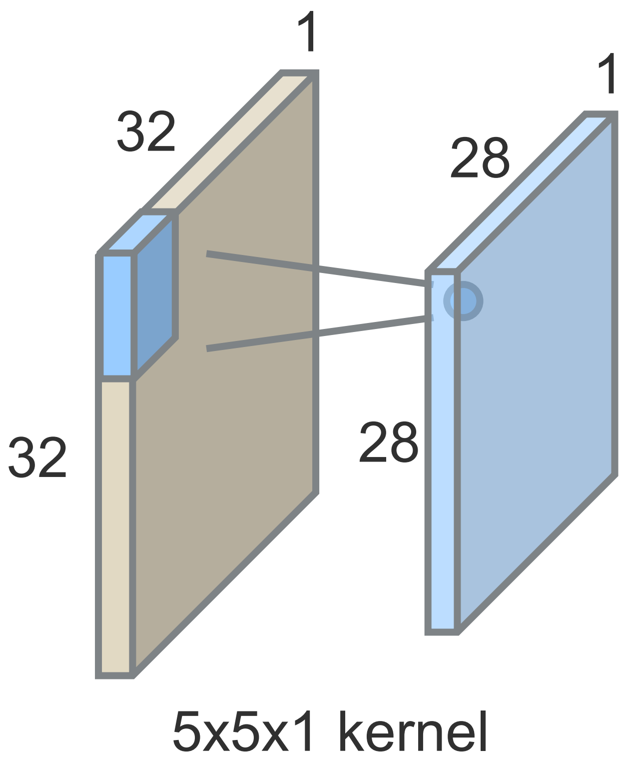
Convolution

- Spatial filter kernel to process (weight) image region, slide over image
- Corresponds to dot product for each output element, same weights for all pixels



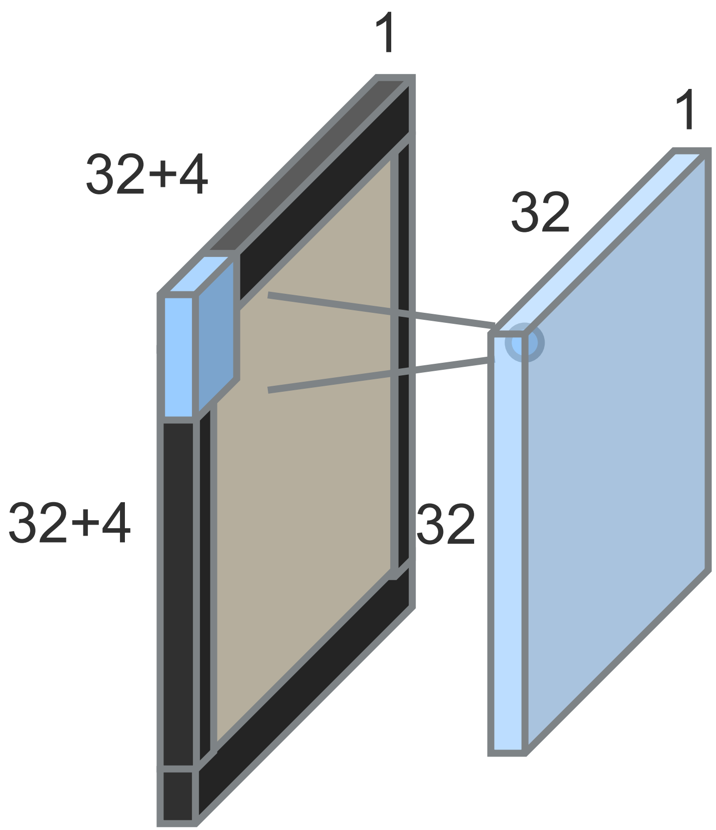
Convolution

- Spatial filter kernel to process (weight) image region, slide over image
- Corresponds to dot product for each output element, same weights for all pixels
- Yields output image (activation layer)



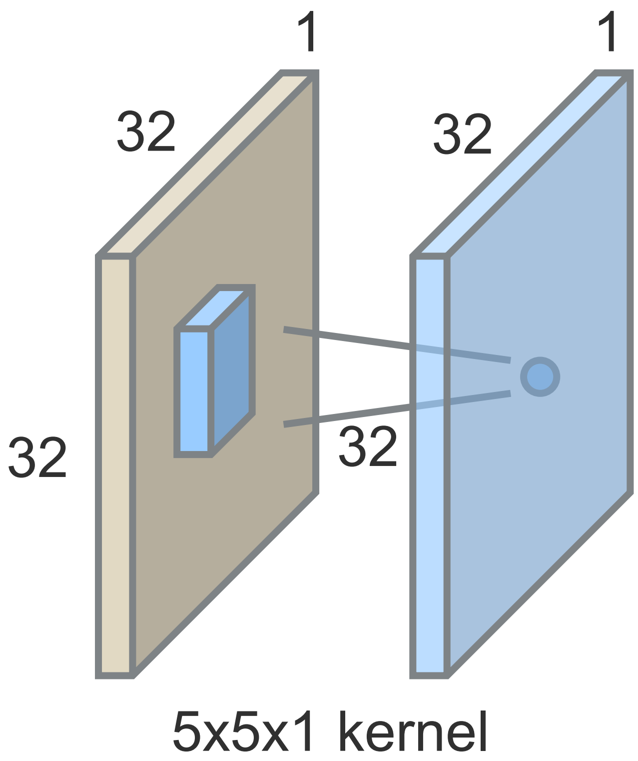
Convolution

- Zero-padding to maintain resolution



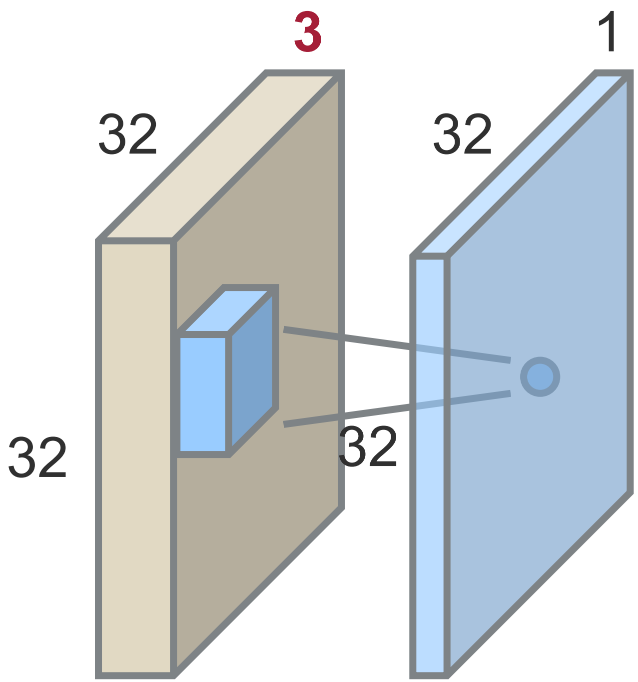
Convolution Layer

- Generalize to multi-channel images and kernels



Convolution Layer

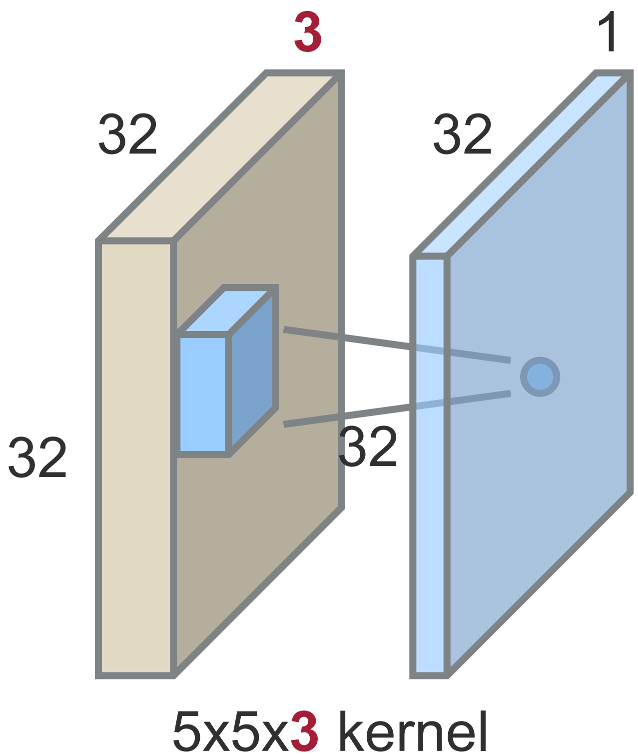
- Generalize to multi-channel images and kernels
- Always filter over all input channels



5x5x**3** kernel

Convolution Layer

- Generalize to multi-channel images and kernels
- Always filter over all input channels

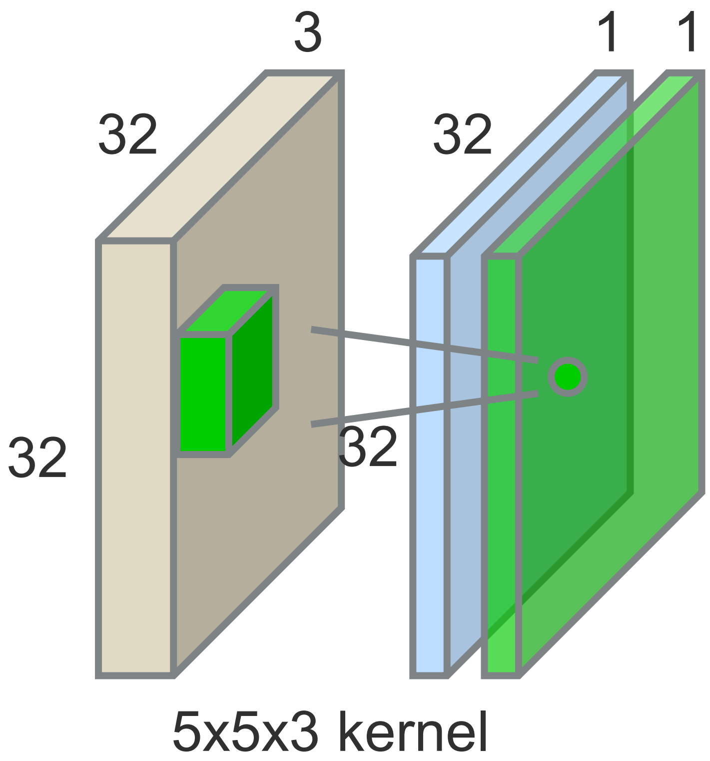


1 ouput number:
dot product between filter
and a 5x5x3 chunk of the image

$$w^T x + b$$

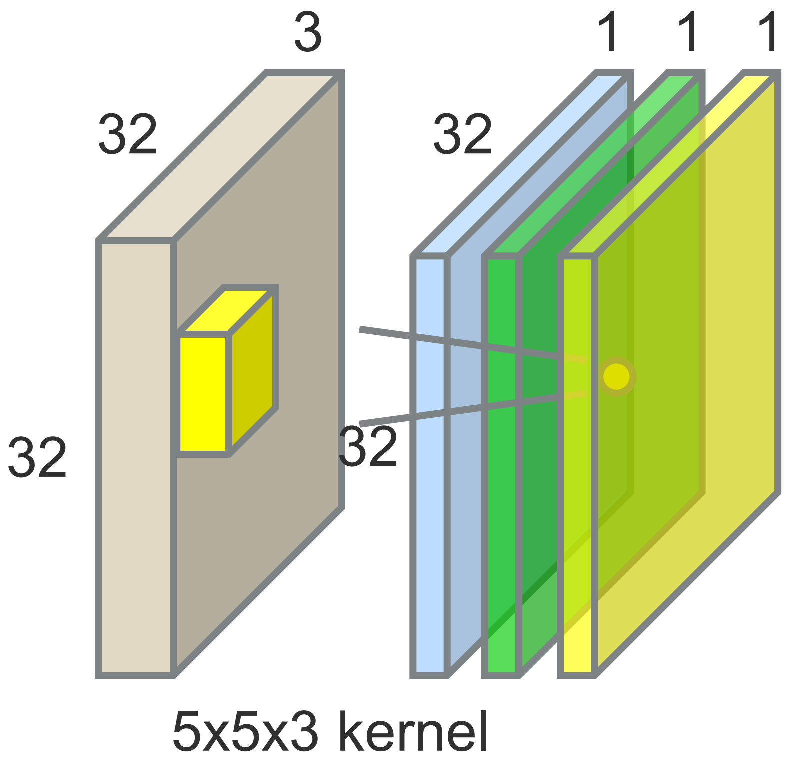
Convolution Layer

- Generalize to multi-channel images and kernels
- Always filter over all input channels
- Multiple output channels



Convolution Layer

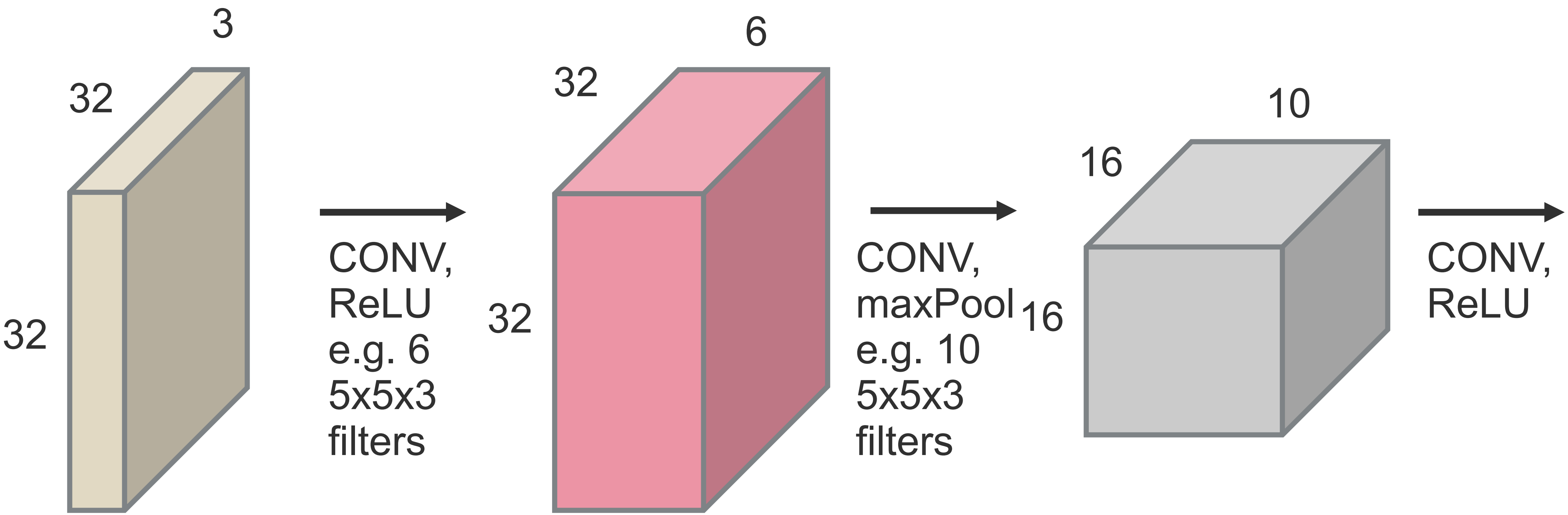
- Generalize to multi-channel images and kernels
- Always filter over all input channels
- Multiple output channels



Fully specified filter kernel:
fourth-order tensor
3x5x5x3

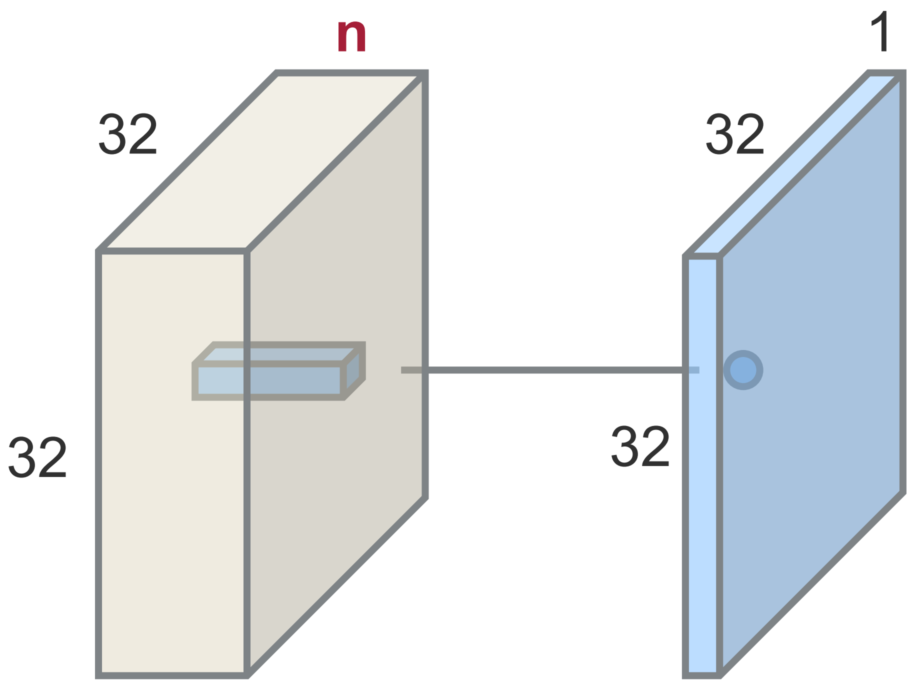
ConvNet

- Sequence of convolution layers, activation functions, … and some other layers



1x1 Convolution Layer

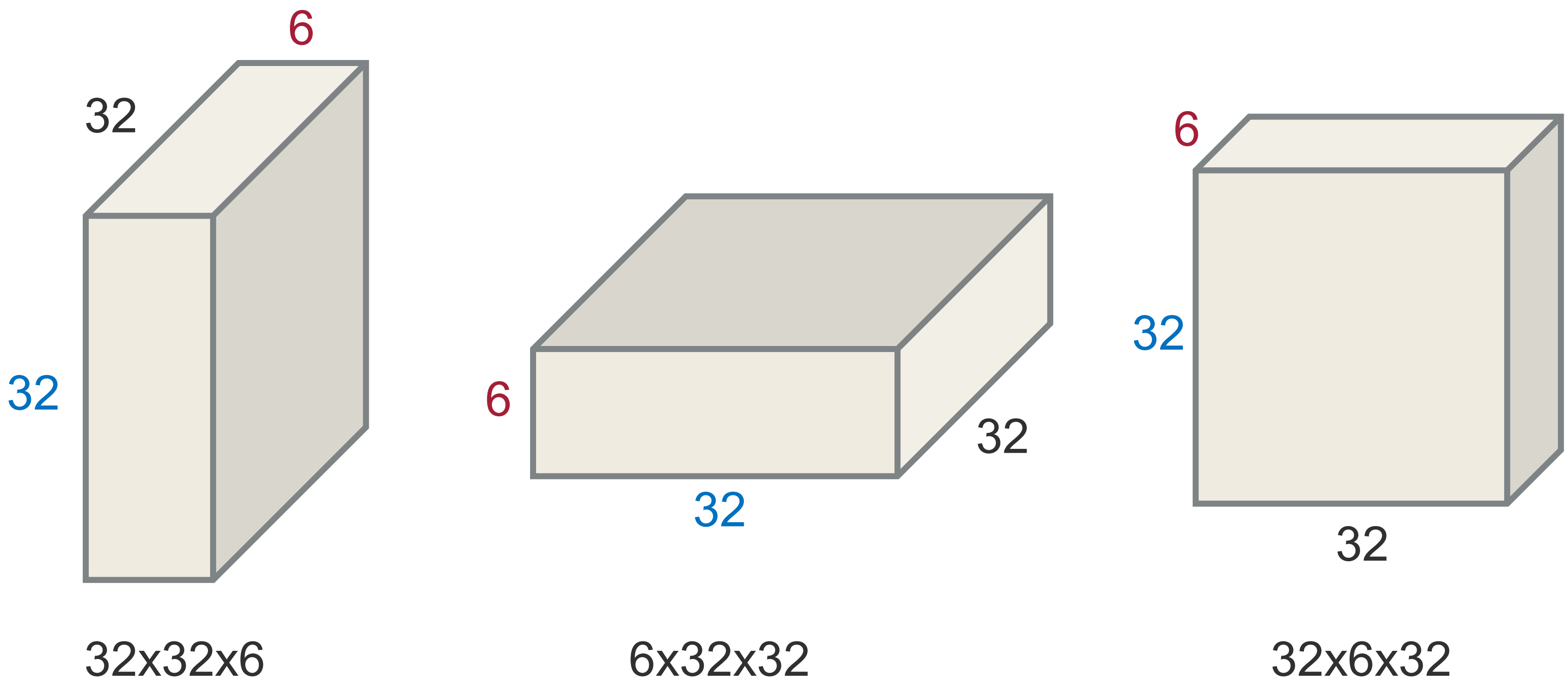
- Kernel size: 1x1xn
- Only compute a weighted sum over the full feature length
- Keeps original resolution



1x1x**n** kernel

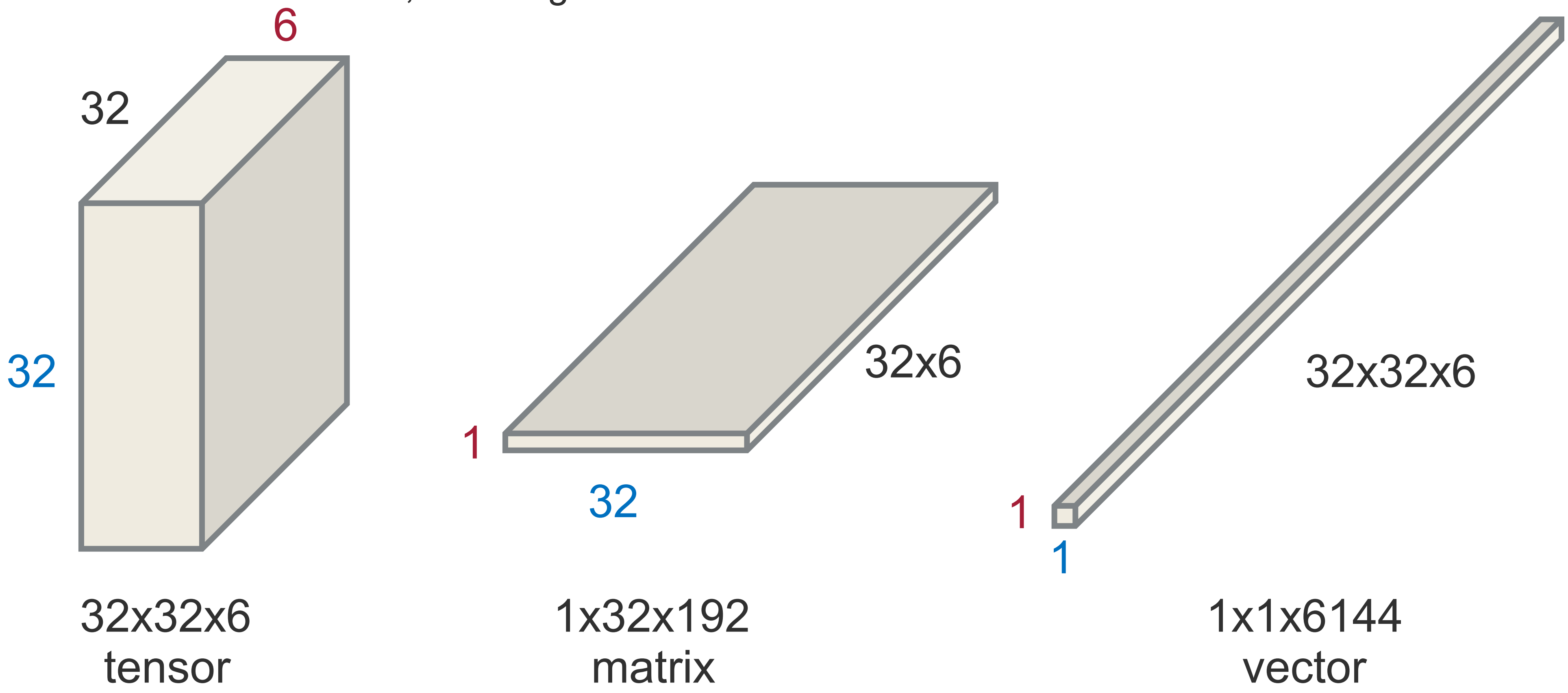
Tensor – Reshape

- Keep the data, just put it into a different shape
- Reorder the dimensions



Tensor – Reshape - Flatten

- Keep the data, just put it into a different shape
- Reorder the dimensions, rearrange



Convolution

$$\begin{bmatrix} O_{11} & O_{12} \\ O_{21} & O_{22} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \right)$$

$$O_{11} = F_{11}X_{11} + F_{12}X_{12} + F_{21}X_{21} + F_{22}X_{22}$$

$$O_{12} = F_{11}X_{12} + F_{12}X_{13} + F_{21}X_{22} + F_{22}X_{23}$$

$$O_{21} = F_{11}X_{21} + F_{12}X_{22} + F_{21}X_{31} + F_{22}X_{32}$$

$$O_{22} = F_{11}X_{22} + F_{12}X_{23} + F_{21}X_{32} + F_{22}X_{33}$$

[Forward And Backpropagation in Convolutional Neural Network – Sujit Rai]

Convolution – Back Propagation

- Back propagations needs to propagate error to original input pixels
- Amounts to convolution by rotated filter kernel $(180°)$

$$\begin{bmatrix} \partial E/\partial X_{11} & \partial E/\partial X_{12} & \partial E/\partial X_{13} \\ \partial E/\partial X_{21} & \partial E/\partial X_{22} & \partial E/\partial X_{23} \\ \partial E/\partial X_{31} & \partial E/\partial X_{32} & \partial E/\partial X_{33} \end{bmatrix} = \text{Full\_Convolution}\left(\begin{bmatrix} \partial E/\partial O_{11} & \partial E/\partial O_{12} \\ \partial E/\partial O_{21} & \partial E/\partial O_{22} \end{bmatrix}, \begin{bmatrix} F_{22} & F_{21} \\ F_{12} & F_{11} \end{bmatrix} \right)$$

$\vdots$

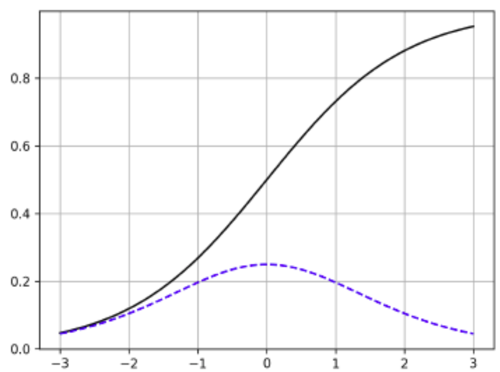
$$\frac{\partial E}{\partial X_{22}} = \frac{\partial E}{\partial Q_{11}} F_{22} + \frac{\partial E}{\partial Q_{12}} F_{21} + \frac{\partial E}{\partial Q_{21}} F_{12} + \frac{\partial E}{\partial Q_{22}} F_{11}$$

$\vdots$

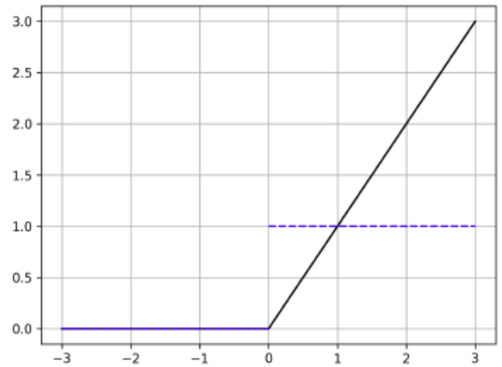
[Forward And Backpropagation in Convolutional Neural Network – Sujit Rai]

Activation Functions

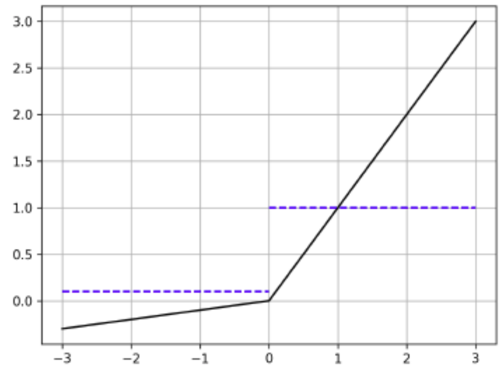
- Signalling when the unit is active: many different choices
- Sigmoid: $f(x) = 1/(1 + \exp(-x))$
  - smooth, problem of saturation, deminishing gradients
- ReLU: $f(x) = \max(0, x)$
  - rectified linear unit: negative range set to zero, otherwise constant grandient
  - No propagation for negative activation → no training / backprop
  - Computationally very efficient
- Leaky ReLU: $f(x) = \max(\alpha x, x)$
  - here with different slope ($\alpha = 0.01$) on the negative side, avoids dead neurons



sigmoid



ReLU



leaky ReLU

Cost-Functions

Given N samples x_i with known label $\hat{y}_i$
Mimimize the distance of the prediction $y(x_i)$ to the label

- MSE – mean squared error

$$C = \frac{1}{N}\sum^{N}[\hat{y}_i - y(x_i)]^2$$

- MAE – mean absolute error

$$C = \frac{1}{N}\sum^{N}|\hat{y}_i - y(x_i)|$$

- Cross-Entropy – miminize negative log likelihood
  - Particularly useful to prevent slow learning with sigmoid activation functions

$$C = -\frac{1}{N}\sum^{N}[\hat{y}_i \ln y(x_i) + (1-\hat{y}_i)\ln(1-y(x_i))]$$

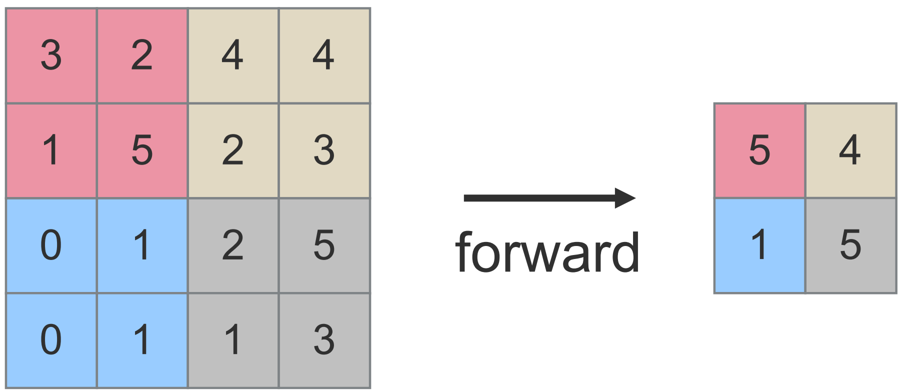
Max Pooling

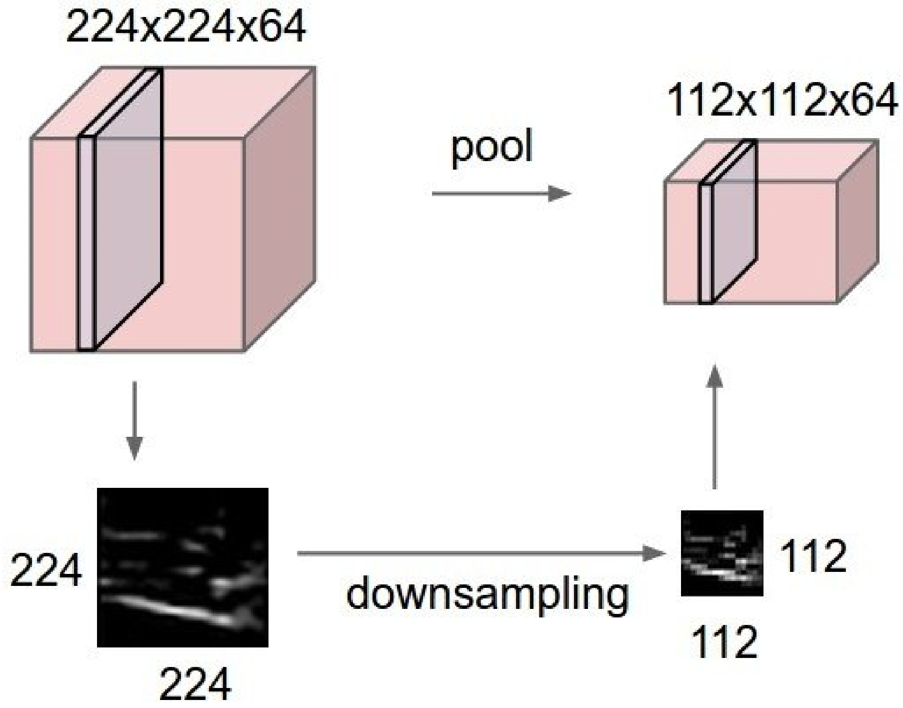
- Goal: reduce the resolution of an activation layer
- „winner“ takes all for a 2x2 region

3	2	4	4
1	5	2	3
0	1	2	5
0	1	1	3

Max Pooling

- Goal: reduce the resolution of an activation layer
- „winner“ takes all for a 2x2 region (stride 2)

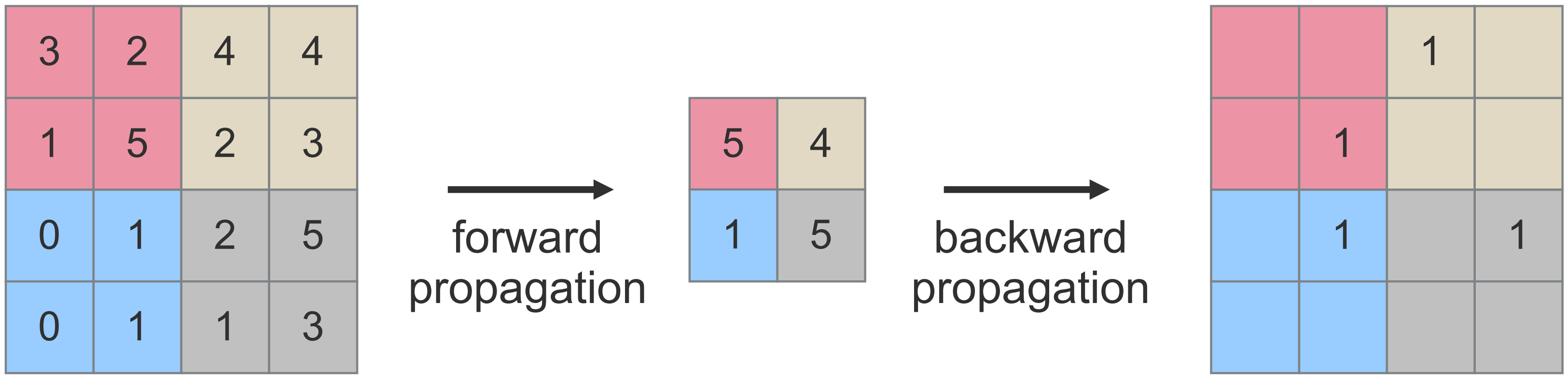




[from Stanford CS231n]

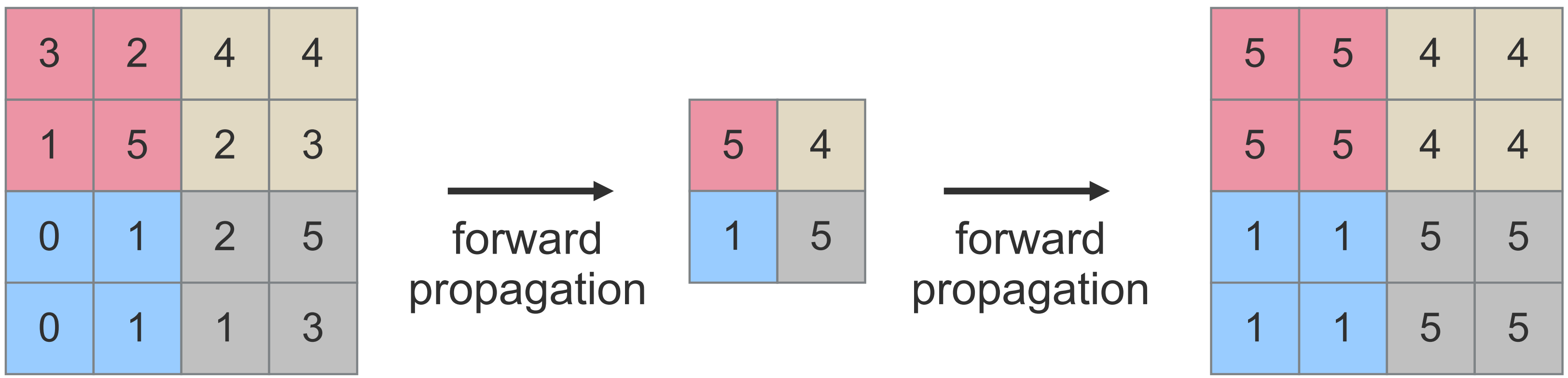
Max Pooling

- Goal: reduce the resolution of an activation layer
- „winner“ takes all for a 2x2 region (stride 2)
- backward pass will be sparse



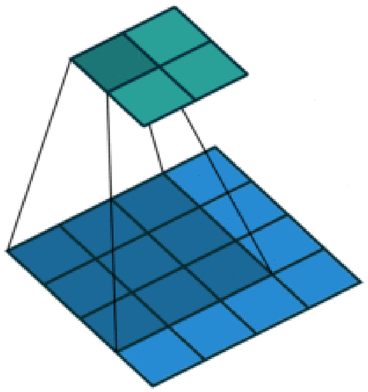
Un-Pooling

- Goal: increase the resolution of an activation layer

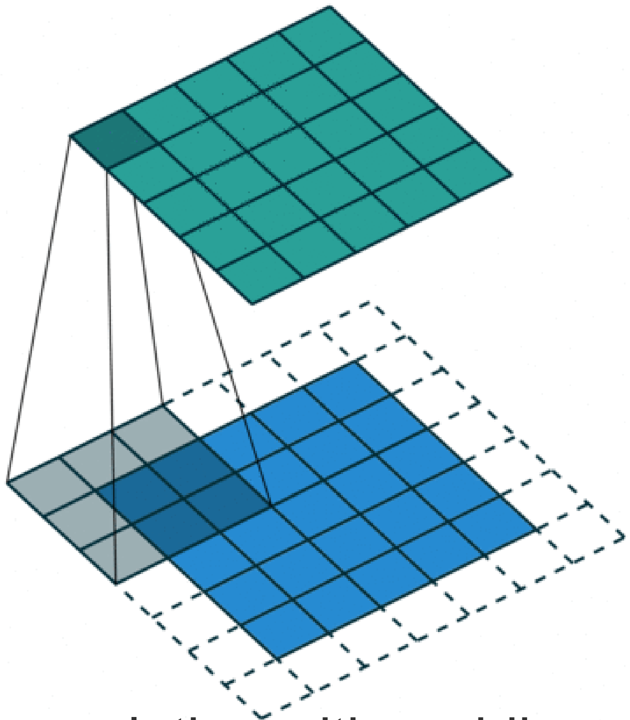


Strided Convolution

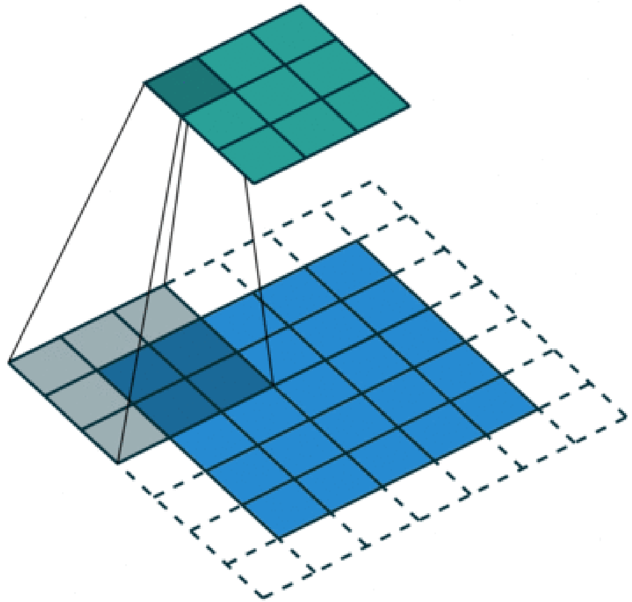
- Generates a weighted output for a 2x2 region (stride 2)
- Blue: input, green: ouput



convolution



convolution with padding

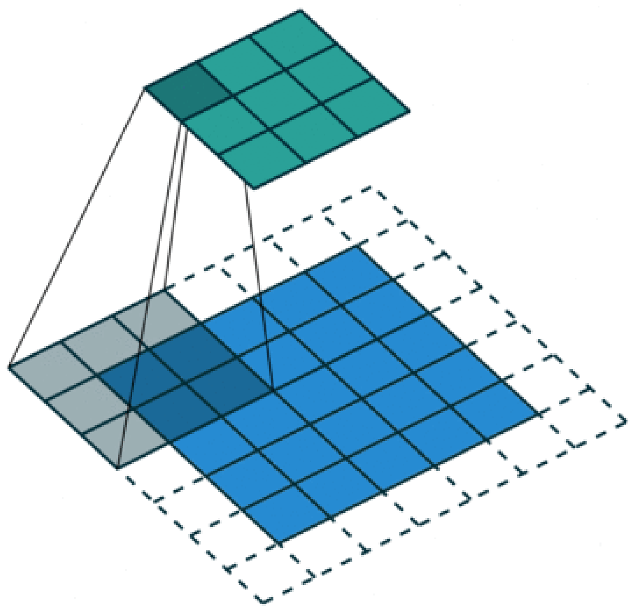


convolution with stride 2

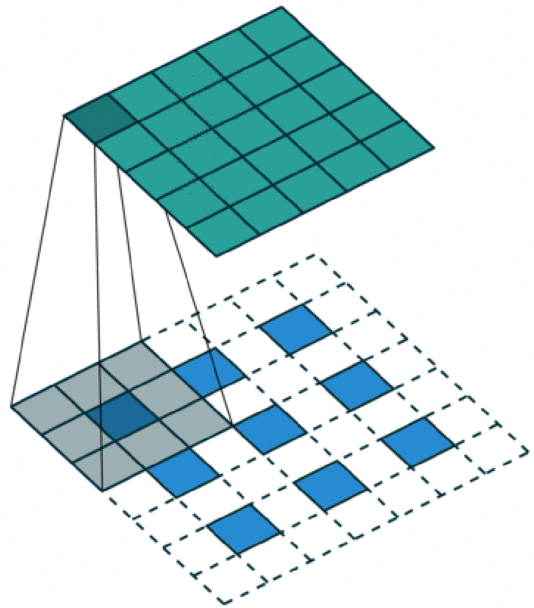
[https://github.com/vdumoulin/conv_arithmetic]

Strided Convolution – Up-Convolution

- Generates a weighted output for a 2x2 region (stride 2)
- Sometimes wrongly called de-convolution



convolution with stride 2
blue: input, green: ouput

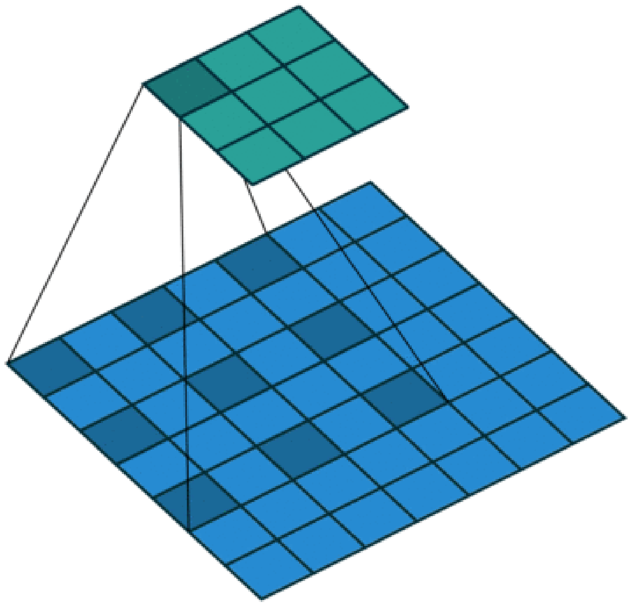


strided transposed convolution
blue: input, green: ouput

[https://github.com/vdumoulin/conv_arithmetic]

Dilated Convolution – aka Á-Trous (with holes)

- Increase the receptive field of a filter kernel without increasing the cost
  - Dilated 5x5, look at every other sample - a cost of 3x3



[https://github.com/vdumoulin/conv_arithmetic]

Dropout

- Train multiple models at the same time without too big training cost, increase robustness

- During training
  - Randomly disable 50% of all connections
  - These connections then does not contribute in backprob this turn

  - The neural network samples a different architecture on every trial.
  - All architectures shares the the same weights
  - Neuron cannot rely on the presence of a particular other neuron, should produce robust features

- During inference
  - Simply weight all connection by 0,5

Batch Normalization (BN)

- Normalizing input (LeCun et al. 1998, Efficient Backprop)
- BN: normalizing **each layer**, for **each mini-batch**
- Greatly accelerates training
- Less sensitive to initialization
- Improves regularization

[S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015]

Batch Normalization (BN)

$$\rightarrow \boxed{\text{layer}} \rightarrow x \rightarrow \quad \hat{x} = \frac{x - \mu}{\sigma} \quad \rightarrow \quad y = \gamma \hat{x} + \beta$$

- μ: mean of x in mini-batch
- σ: std of x in mini-batch
- γ: scale
- β: shift

- μ, σ: functions of x, analogous to responses
- γ, β: parameters to be learned, analogous to weights

Batch Normalization (BN)

$$\rightarrow \boxed{\text{layer}} \rightarrow x \rightarrow \quad \hat{x} = \frac{x - \mu}{\sigma} \quad \rightarrow \quad y = \gamma \hat{x} + \beta$$

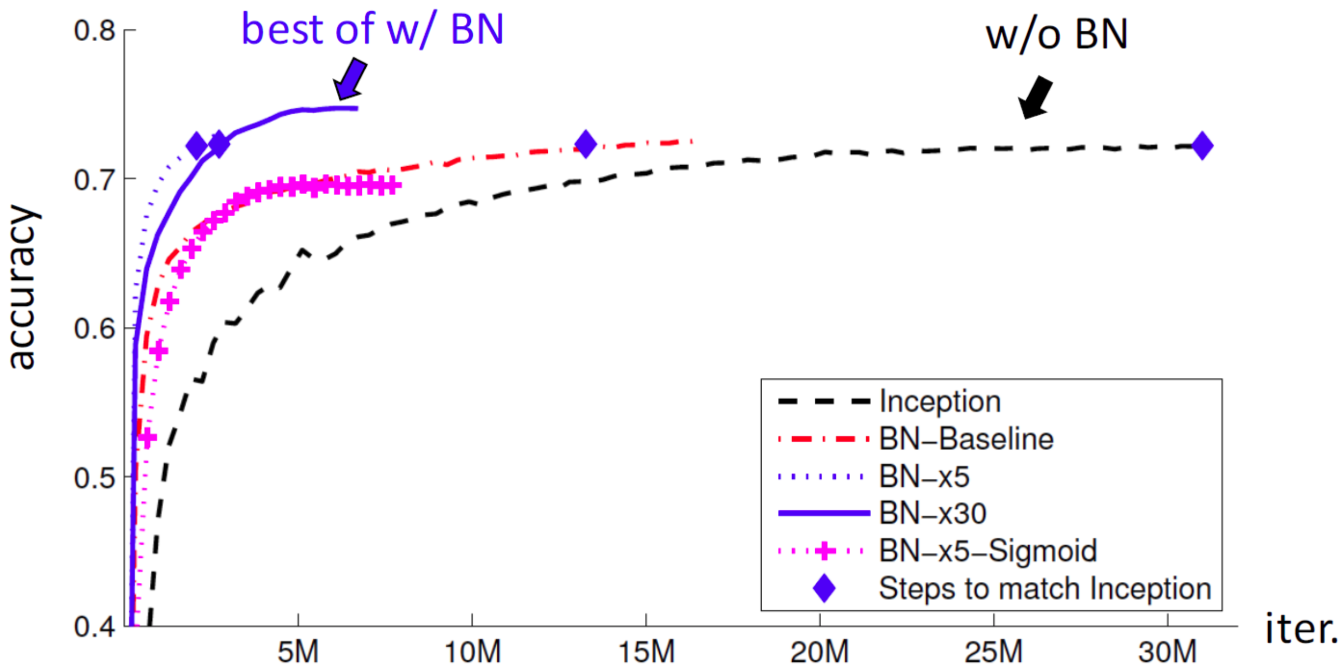
2 modes of BN:

- Train mode:
  - μ, σ are functions of x; backprop gradients
- Test mode:
  - μ, σ are pre-computed* on training set

Caution: make sure your BN is in a correct mode

*: by running average, or post-processing after training

Batch Normalization (BN)

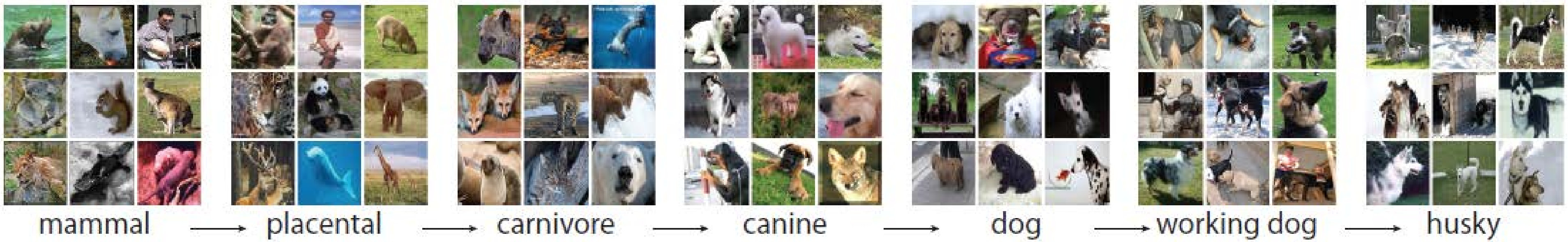


[S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015]

IMAGENET is a knowledge ontology

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

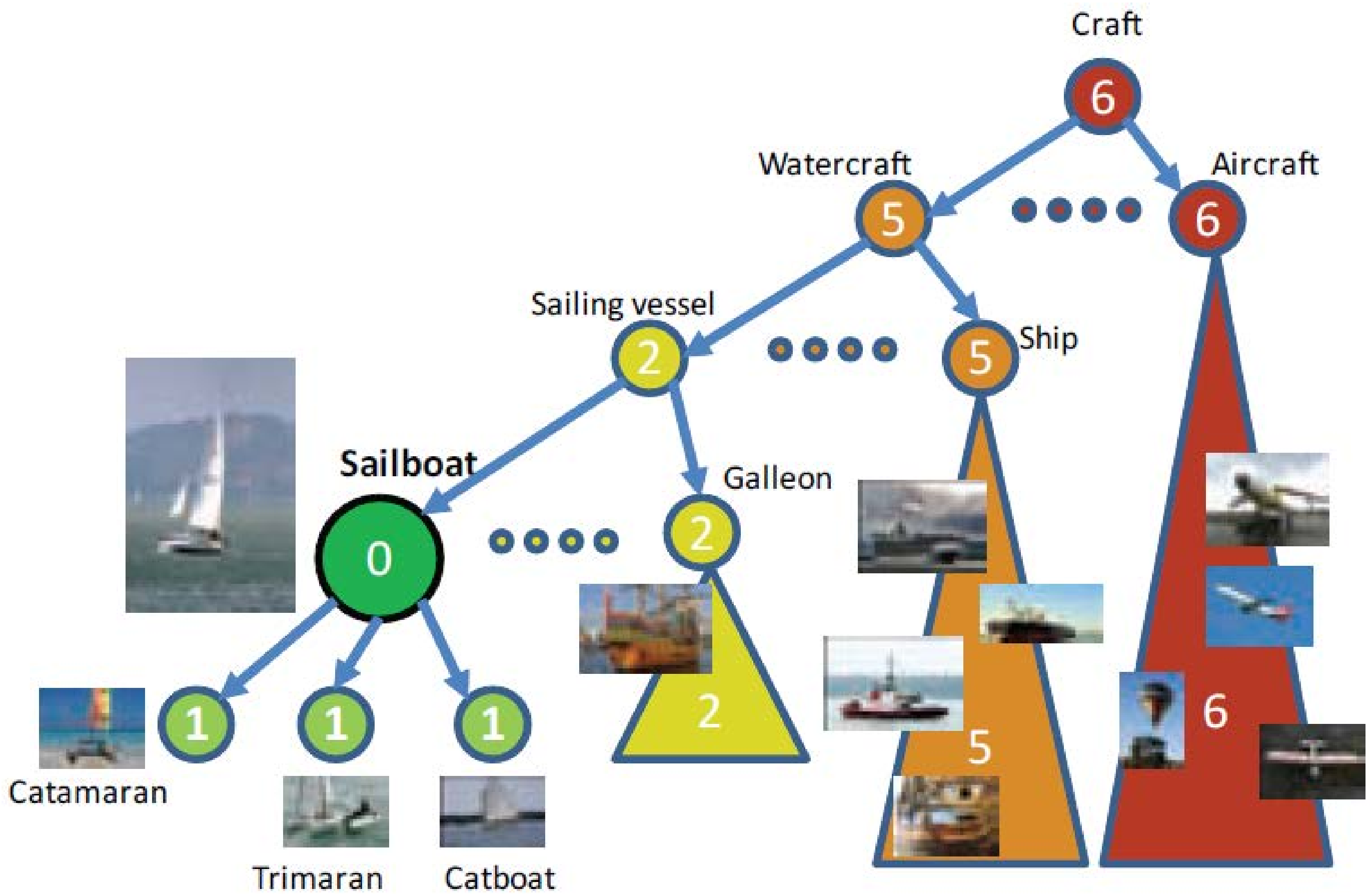
- Taxonomy
- Partonomy
- The “social network” of visual concepts
  - Hidden knowledge and structure among visual concepts
  - Prior knowledge
  - Context



Deng, Berg, Li, & Fei-Fei, *ECCV2010*

Ontology with Hierarchy

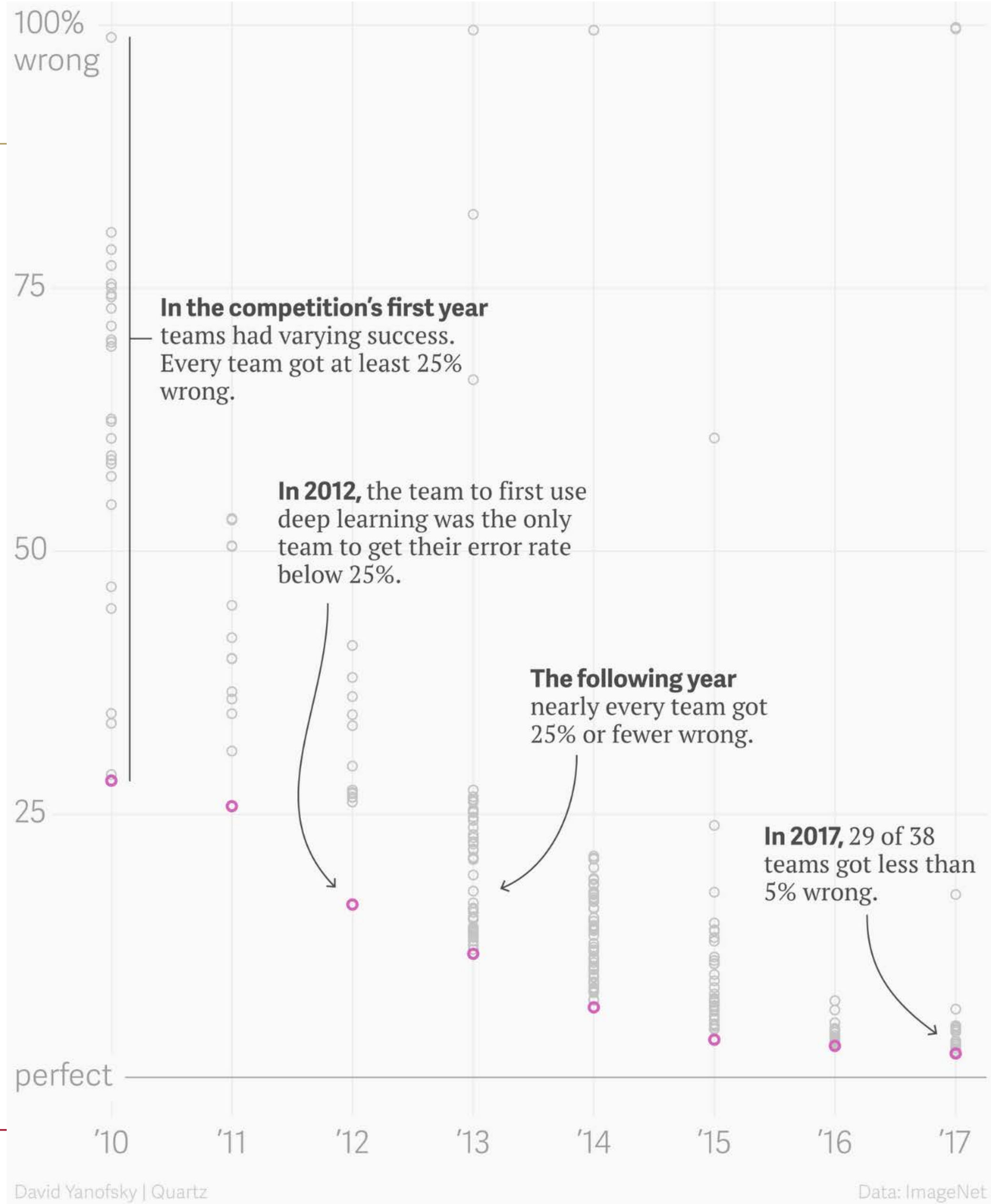
- Datasets have very different “density” or “sparsity”



Deng, Berg, Li, & Fei-Fei, *ECCV2010*

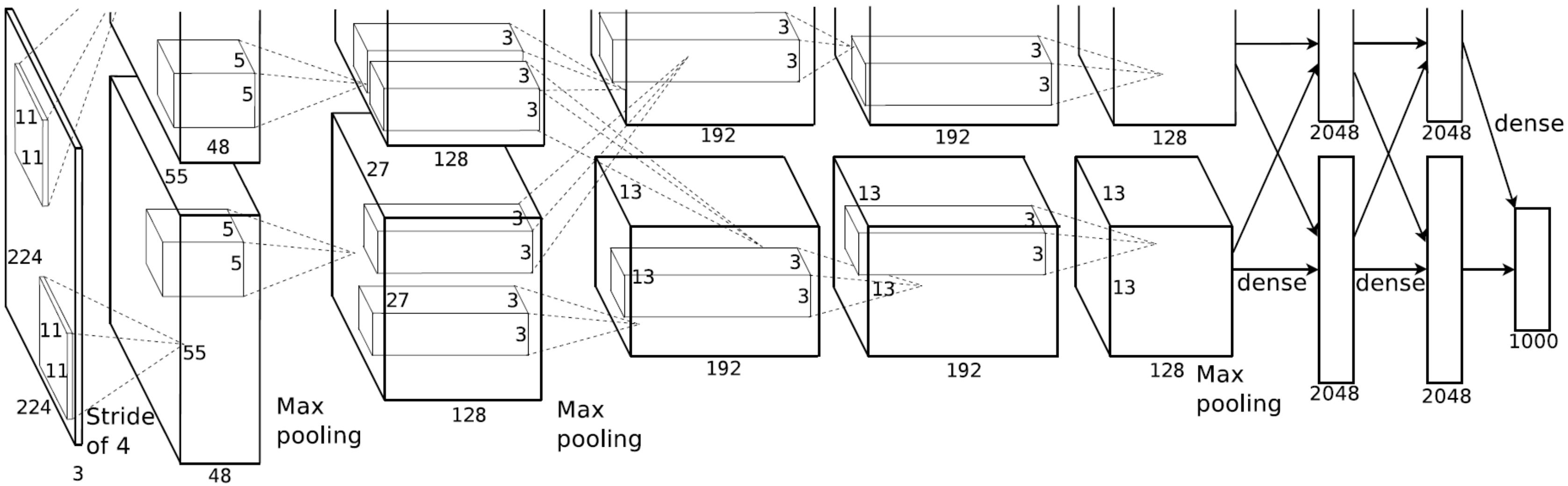
IMAGENET

ILSVRC – ImageNet Large Scale Visual Recognition Competition



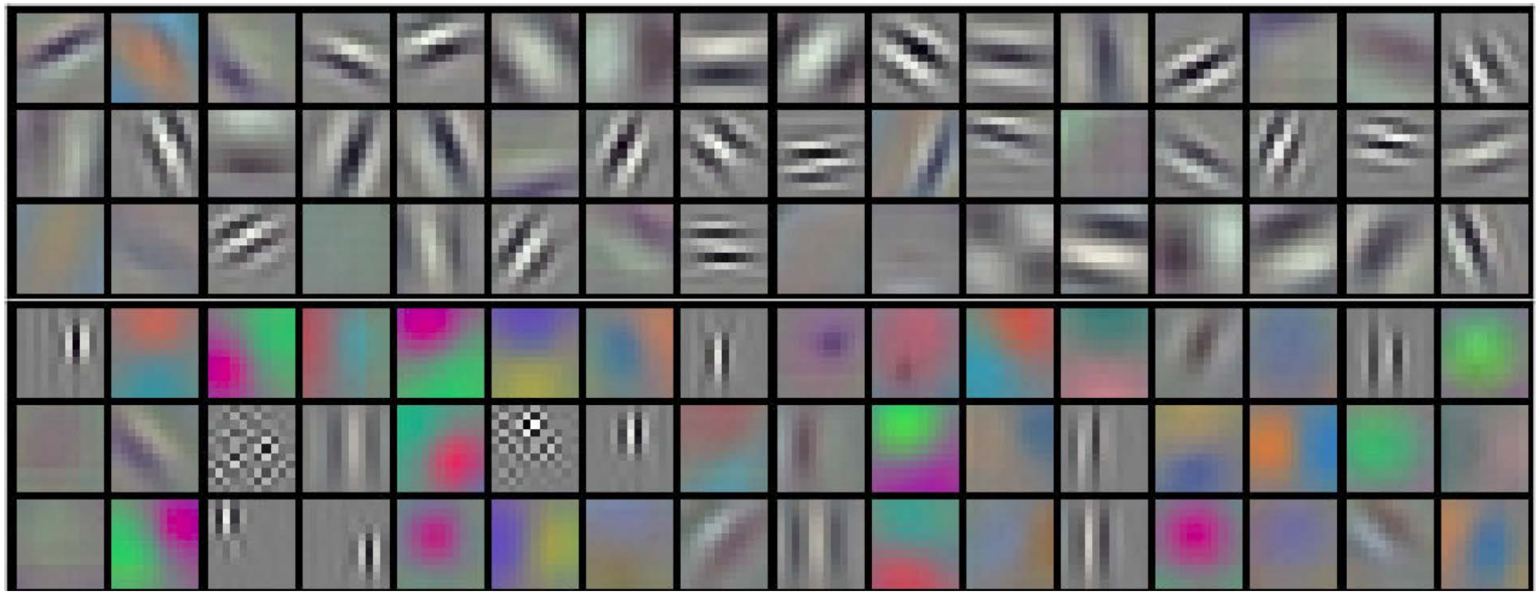
AlexNet

- ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, NIPS 2012 – winner of the ImageNet ILSVRC
- Two parallel subnets trained on two separate GPUs with little interaction



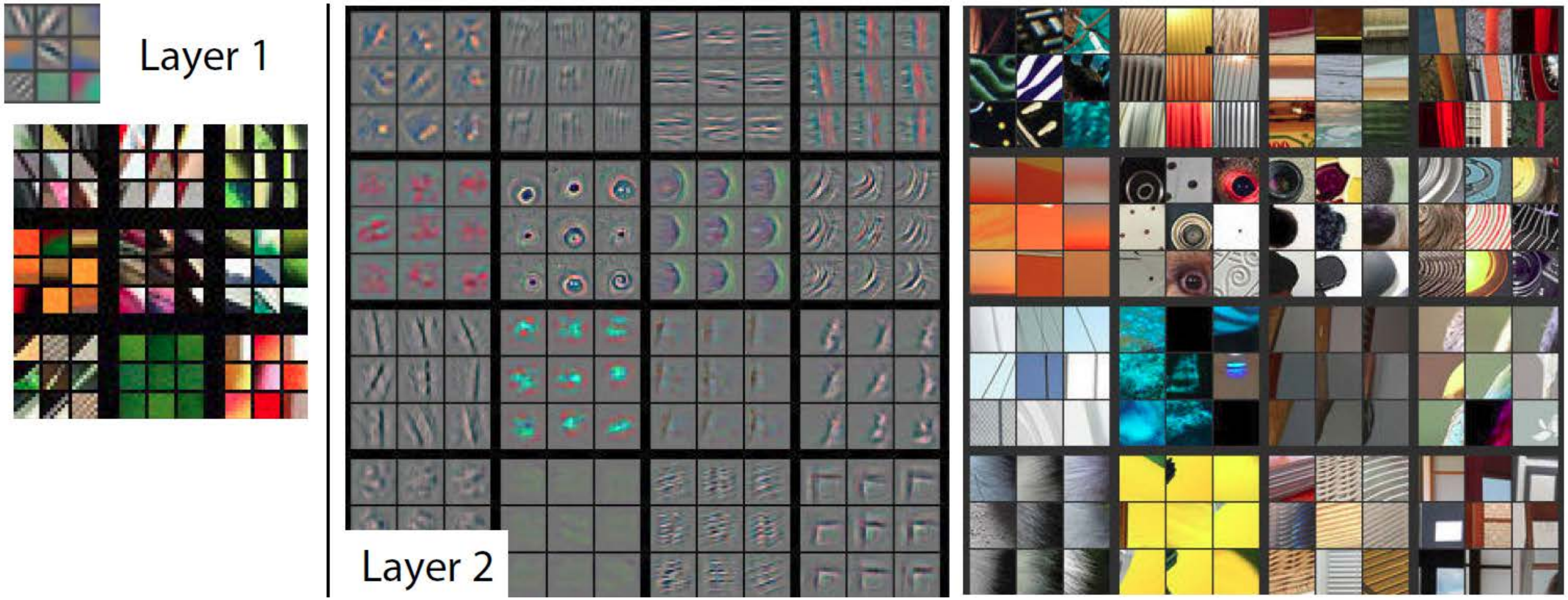
AlexNet

- Resulting features on the first layer
- Similar to response kernels in early human vision



Visualizing Neural Networks

- Matthew D. Zeiler and Rob Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
- Localize the image and feature location that reacted the most to a given kernel – deconvolution network



VGG16 / 19

- Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015 – Winner ILSVRC 2014

- Visual Geometry Group at Oxford

- Main idea
  - Enable deeper networks
  - Propose 3x3 convolutions (even 1x1 convolution)
  - Receptive field can still be large when coupling multiple 3x3 layers
  - Feature length increase after max pooling (pixels vs. expressiveness)
  - Keep it deep. Keep it simple

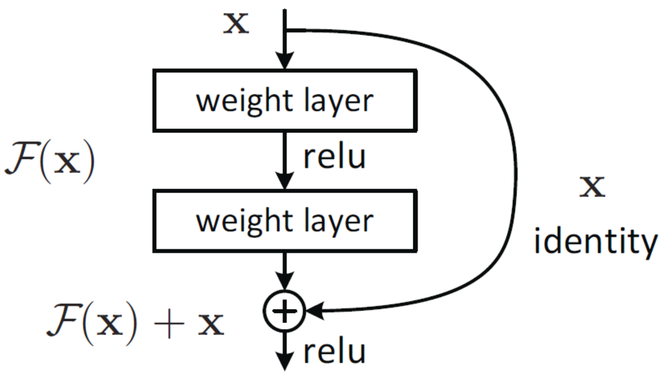
- Good result on classification and localization

VGG 19
conv 3 64
conv 3 64
maxpool
conv 3 128
conv 3 128
maxpool
conv 3 256
conv 3 256
conv 3 256
conv 3 256
maxpool
conv 3 512
conv 3 512
conv 3 512
conv 3 512
maxpool
conv 3 512
conv 3 512
conv 3 512
conv 3 512
maxpool
FC 4096
FC 4096
FC 1024
softmax

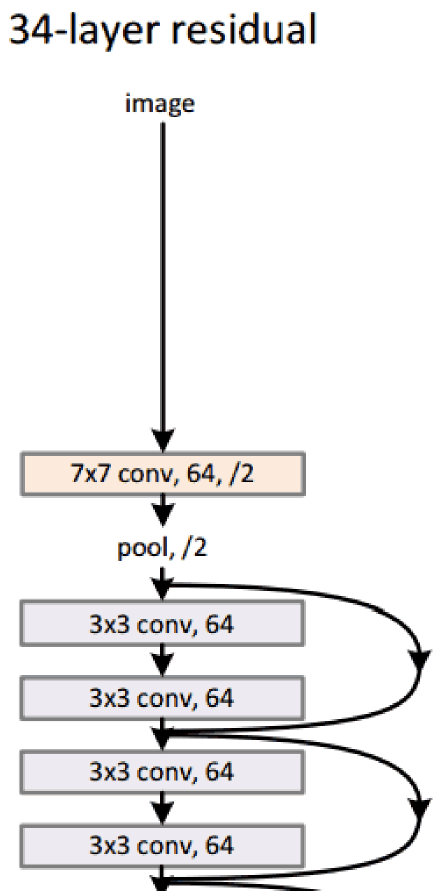
ResNet

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition

- Record number of layers: 152
- Enabled by residual connection
  - Easier to learn a delta than the full signal



- Outperforms humans on recognition



Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)

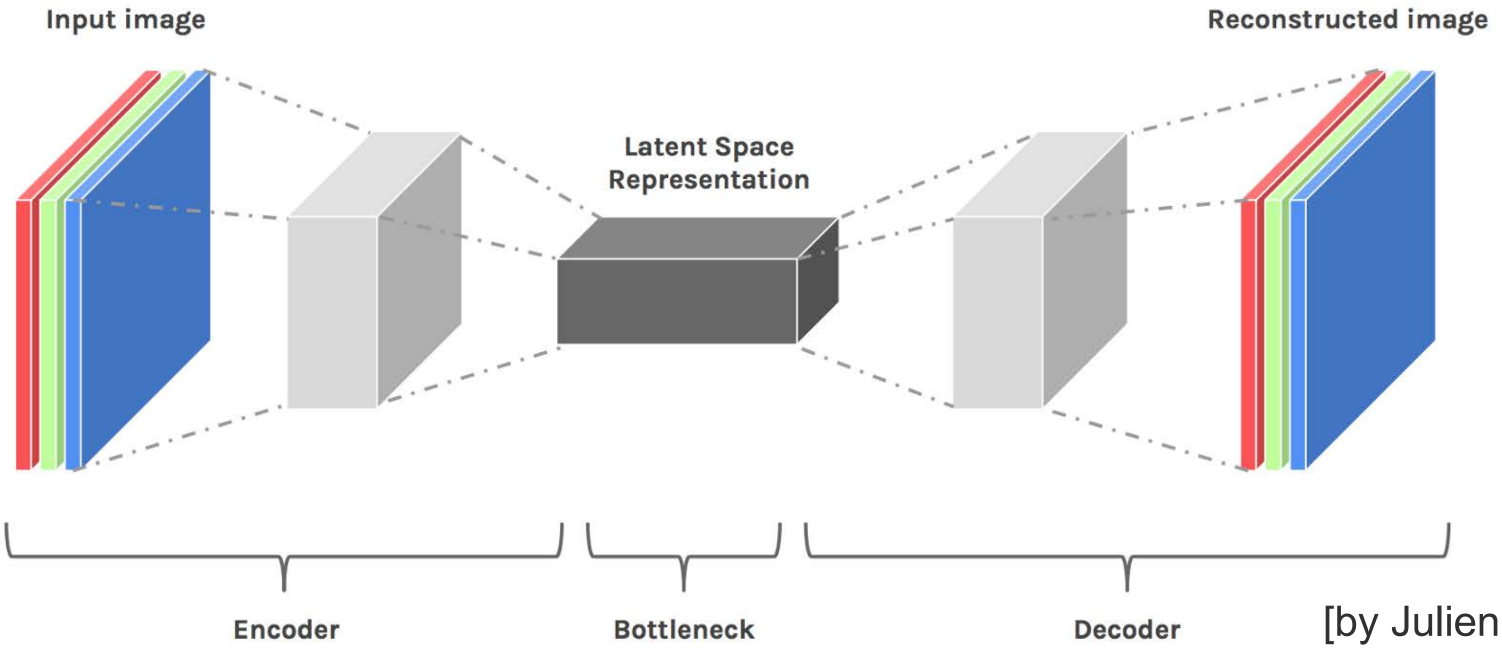
VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. CVPR 2016.

Autoencoder

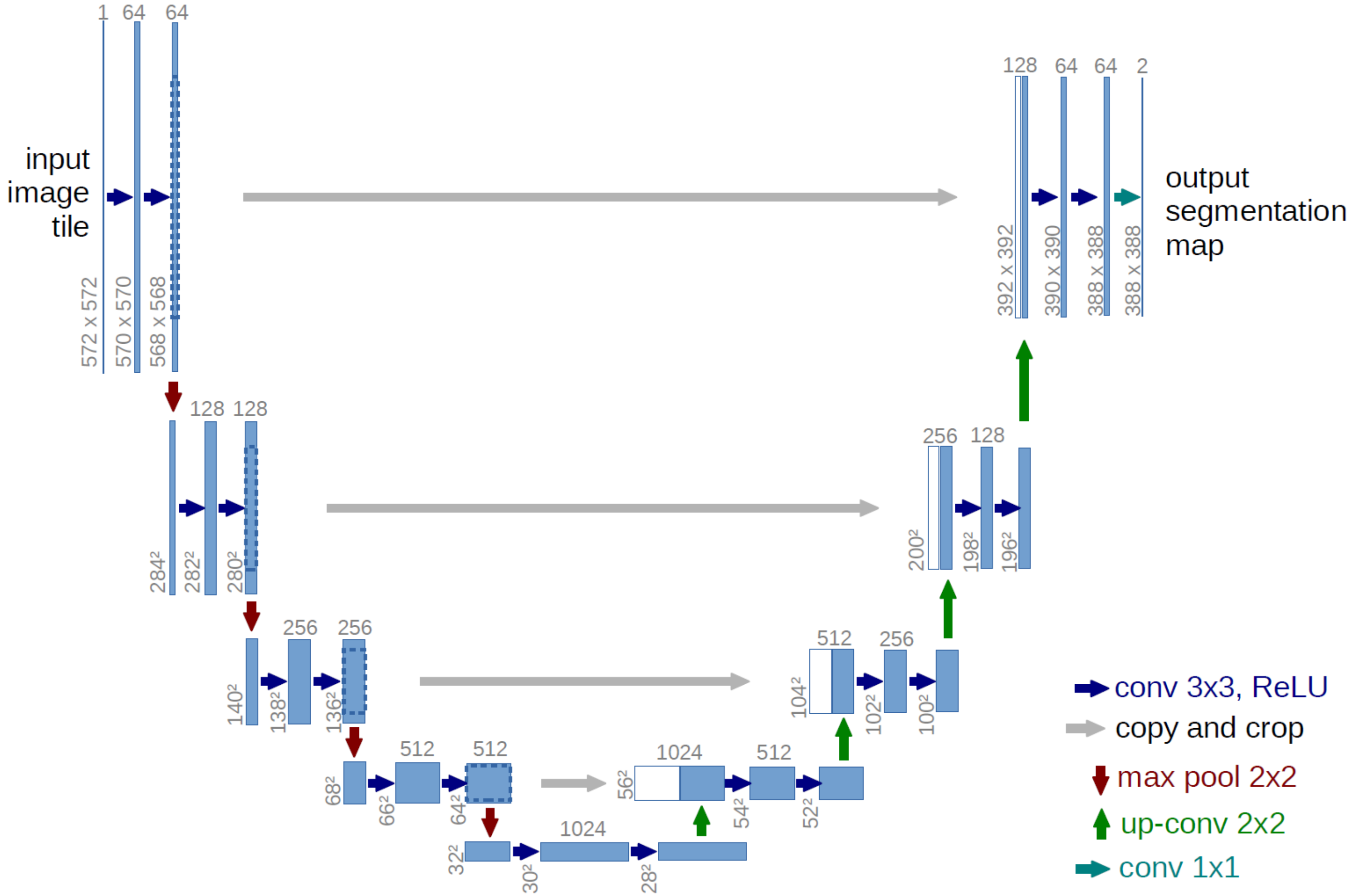
- To process rather than to classify
  - Input image contains stuff (independent of the scene content) that should not be there
  - By introducing a bottleneck the AE is forced to **conenctrate on the significant latent information**
- Compare to PCA / low rank approximation



[by Julien Despois]

U-Net

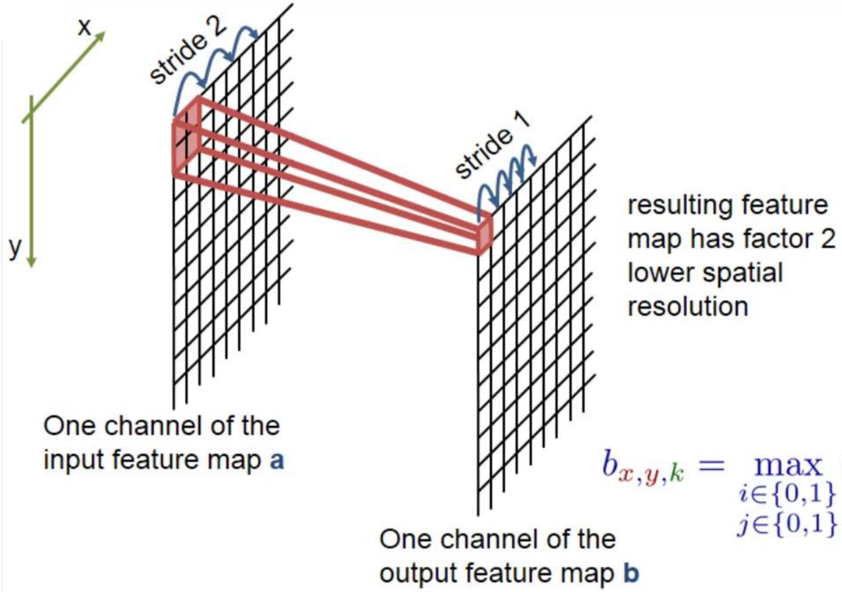
- Autoencoder
- Skip-connections (cmp. ResNet)
- Multi-resolution
  - Extract what
  - Extract where
  - Propagate to full resolution



[Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015]

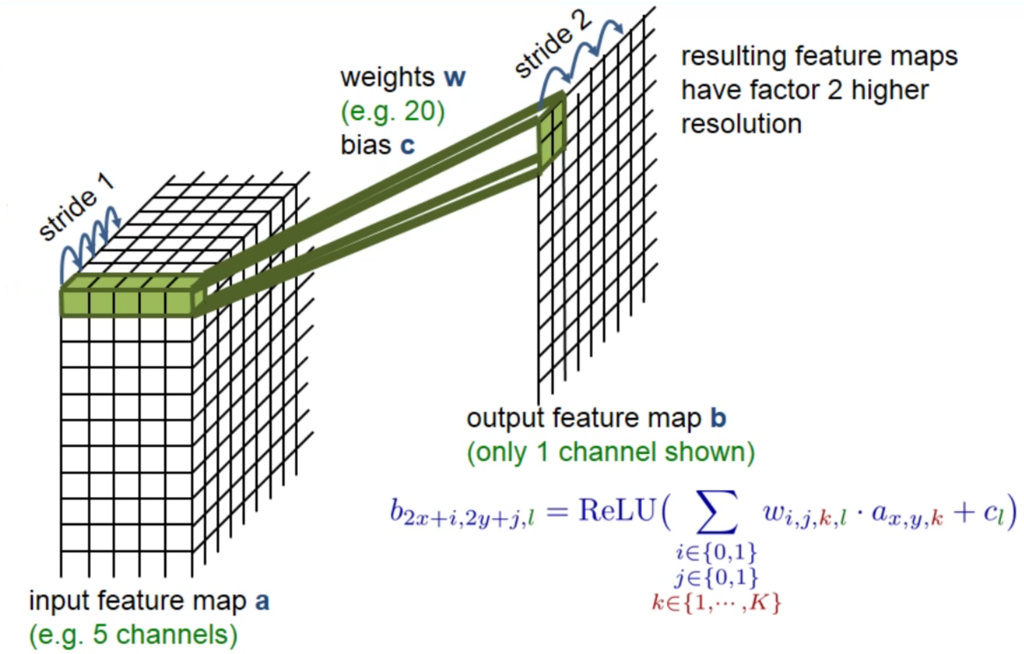
U-Net

- Basic processing steps



$$b_{x,y,k} = \max_{\substack{i\in\{0,1\}\\ j\in\{0,1\}}} (a_{2x+i,2y+j,k})$$

2x2 max pooling



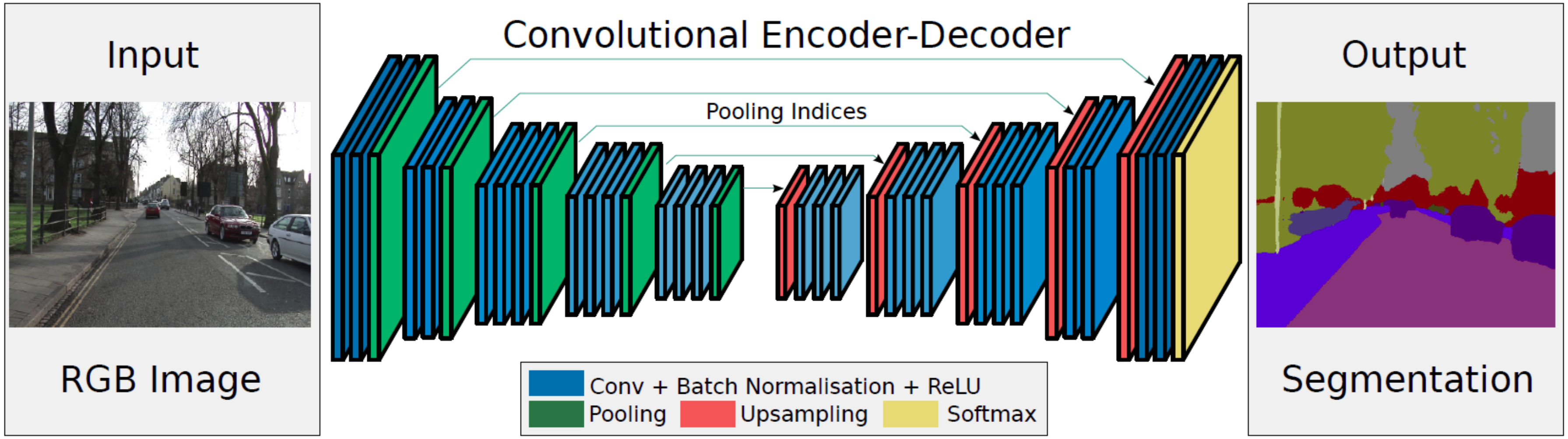
$$b_{2x+i,2y+j,l} = \mathrm{ReLU}\Big(\sum_{\substack{i\in\{0,1\}\\ j\in\{0,1\}\\ k\in\{1,\cdots,K\}}} w_{i,j,k,l} \cdot a_{x,y,k} + c_l \Big)$$

2x2 up-convolution

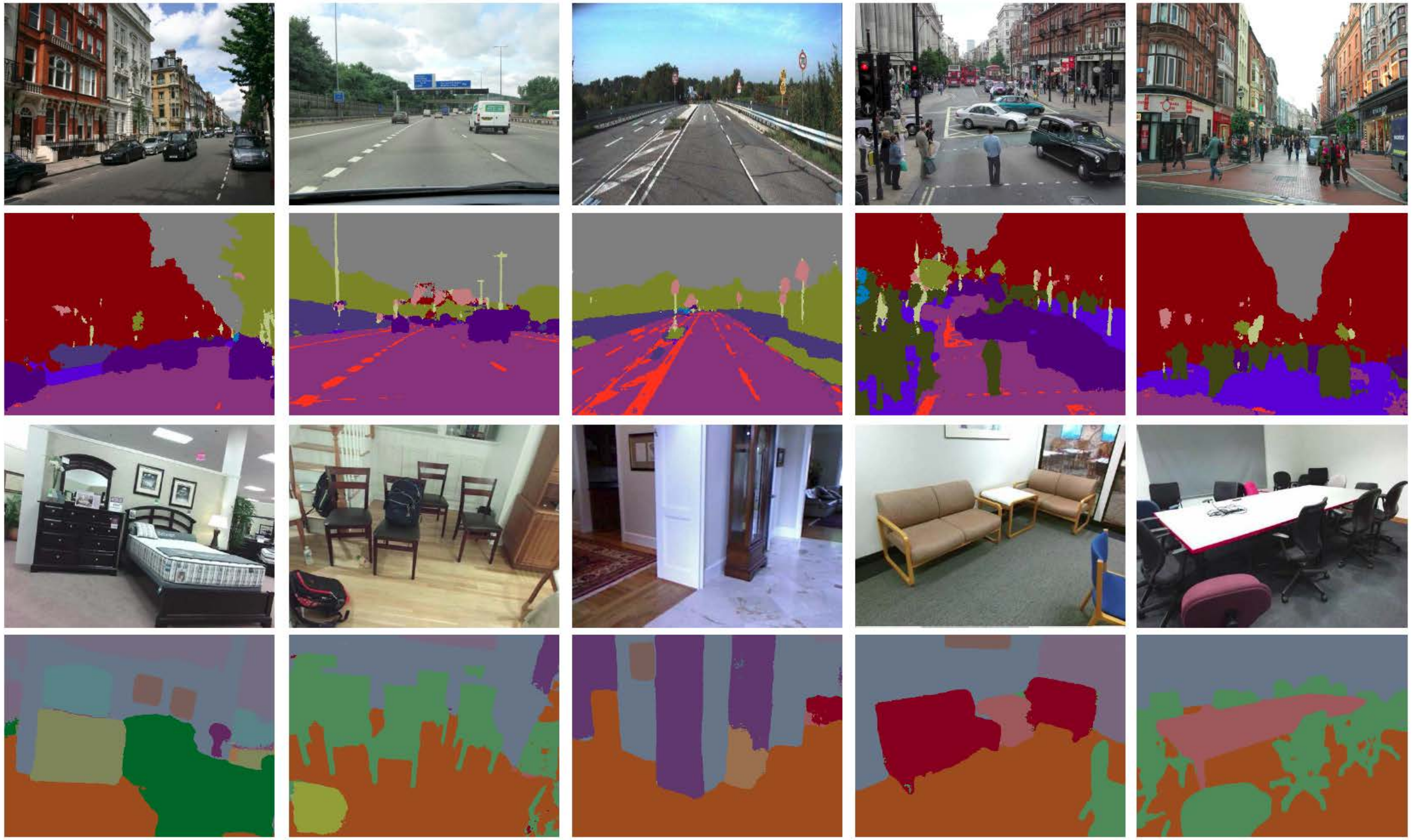
[Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015]

SegNet

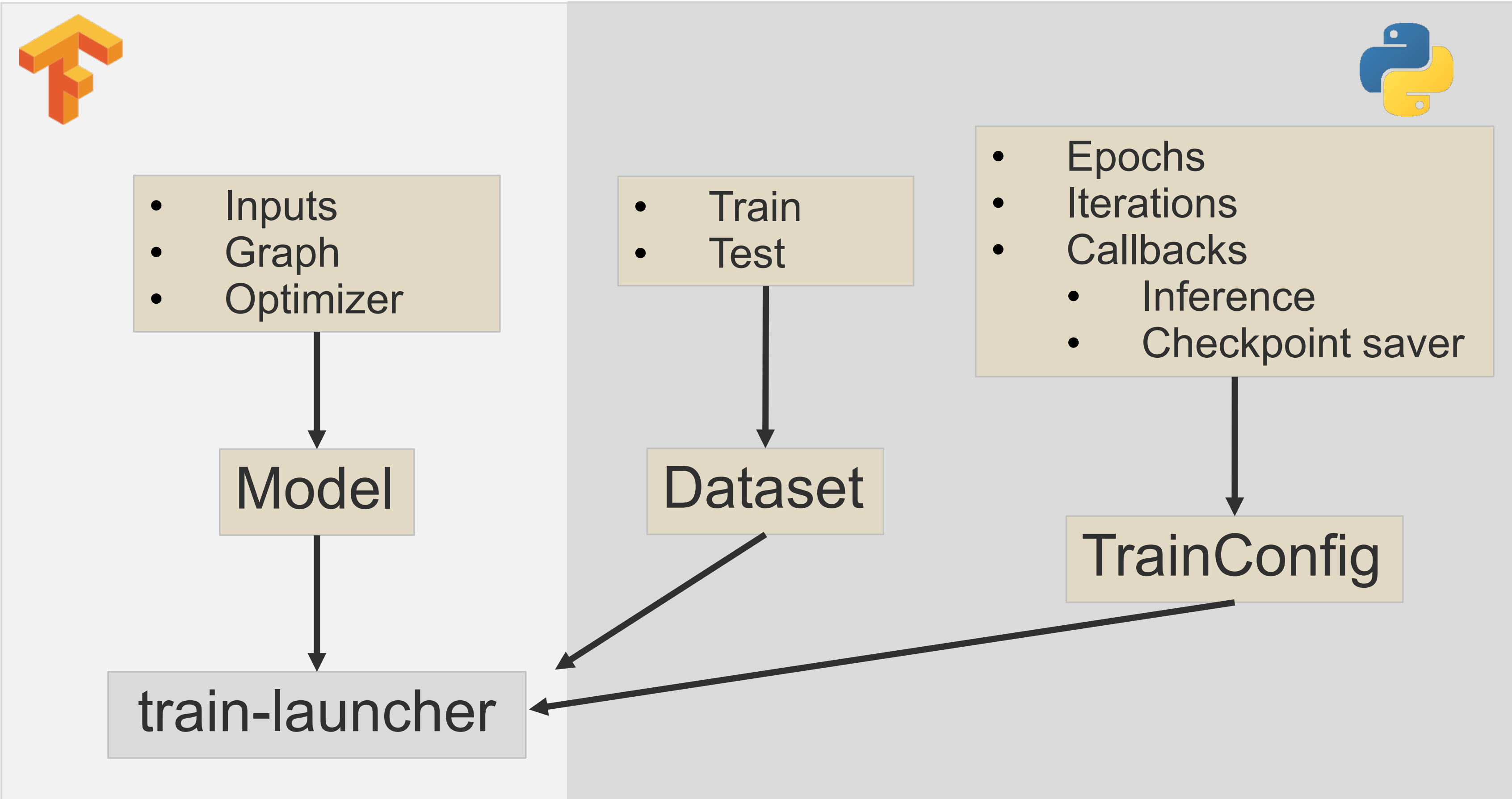
- Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017



SegNet - Results



Training in Tensorpack



Training in Tensorpack

```python
model = Model()

dataset_train = BatchData(dataset.Mnist('train'), 128)
dataset_test = BatchData(dataset.Mnist('test'), 256, remainder=True)


config = TrainConfig(model=model,
    dataflow=dataset_train,
    callbacks=[ ModelSaver()],
    steps_per_epoch=dataset_train.size(),
    max_epoch=100)

trainer = SyncMultiGPUTrainerParameterServer(nr_gpu)
launch_train_with_config(config, trainer)
```

Model

Dataset

TrainConfig

train-launcher

Training in Tensorpack (Model)

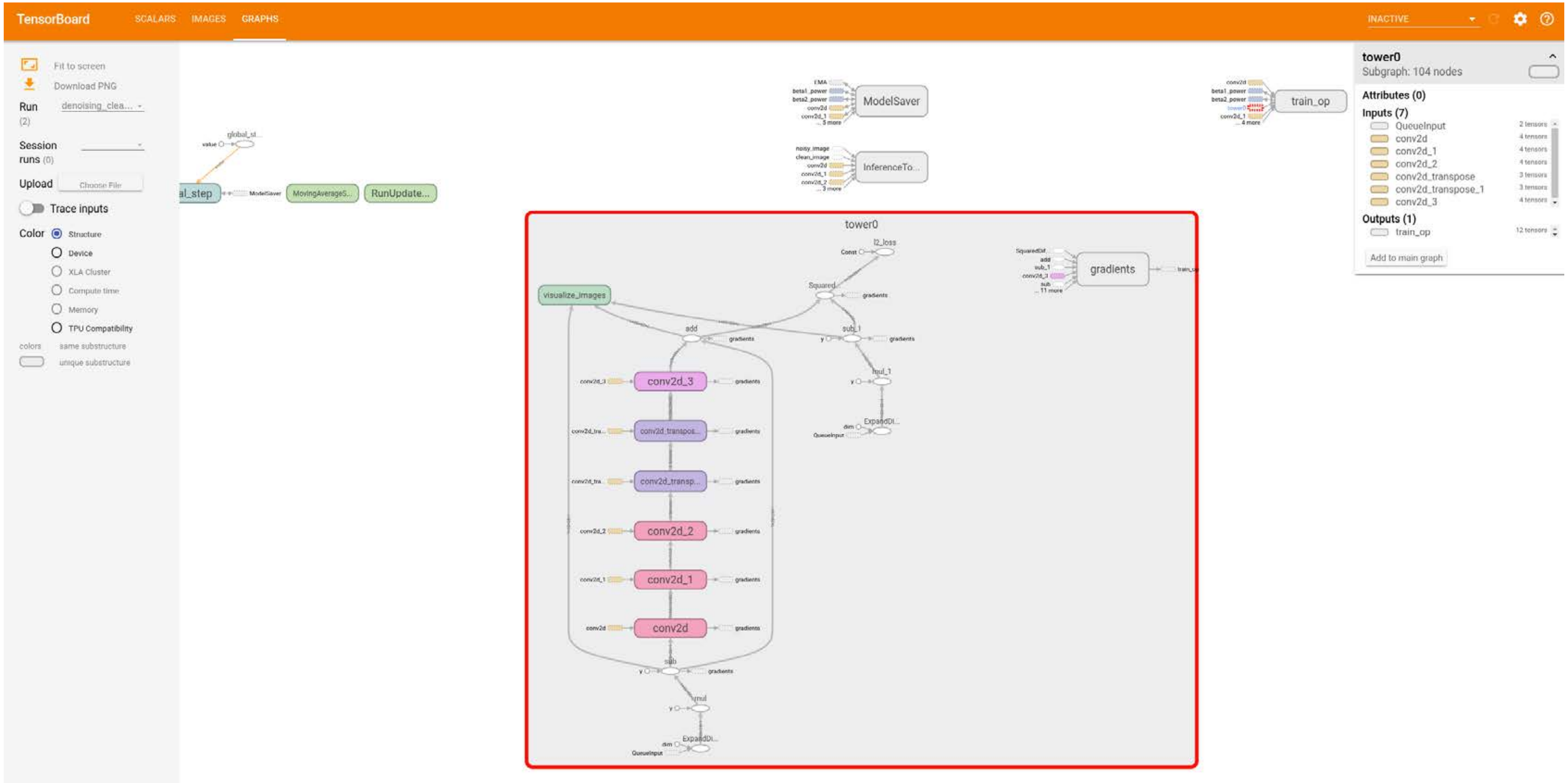
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

```
class Model(ModelDesc):

  def inputs(self):
    return [tf.placeholder(tf.float32, (None, 28, 28), 'image‘),
            tf.placeholder(tf.int32, (None,), 'labels‘)]

  def build_graph(self, image, labels):
    image = tf.expand_dims(image, 3) * 2 - 1
    net = tf.layers.conv2d(image, 32, 3, padding='same‘, activation=tf.nn.relu, name='conv0‘)
    net = tf.layers.max_pooling2d(net, 2, 2, padding='valid')
    net = tf.layers.conv2d(net, 32, 3, padding='same', activation=tf.nn.relu, name='conv1')
    net = tf.layers.conv2d(net, 32, 3, padding='same', activation=tf.nn.relu, name='conv2')
    net = tf.layers.max_pooling2d(net, 2, 2, padding='valid')
    net = tf.layers.conv2d(net, 32, 3, padding='same', activation=tf.nn.relu, name='conv3')
    net = tf.layers.flatten(net)
    net = tf.layers.dense(net, 512, activation=tf.nn.relu, name='fc0‘)
    net = tf.layers.dropout(net, rate=0.5, training=get_current_tower_context().is_training)
    logits = tf.layers.dense(net, 10, activation=tf.identity, name='fc1‘)

    cost = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=logits, labels=labels)
    return tf.reduce_mean(cost, name='cross_entropy_loss‘)

  def optimizer(self):
    return tf.train.AdamOptimizer(1e-3)
```

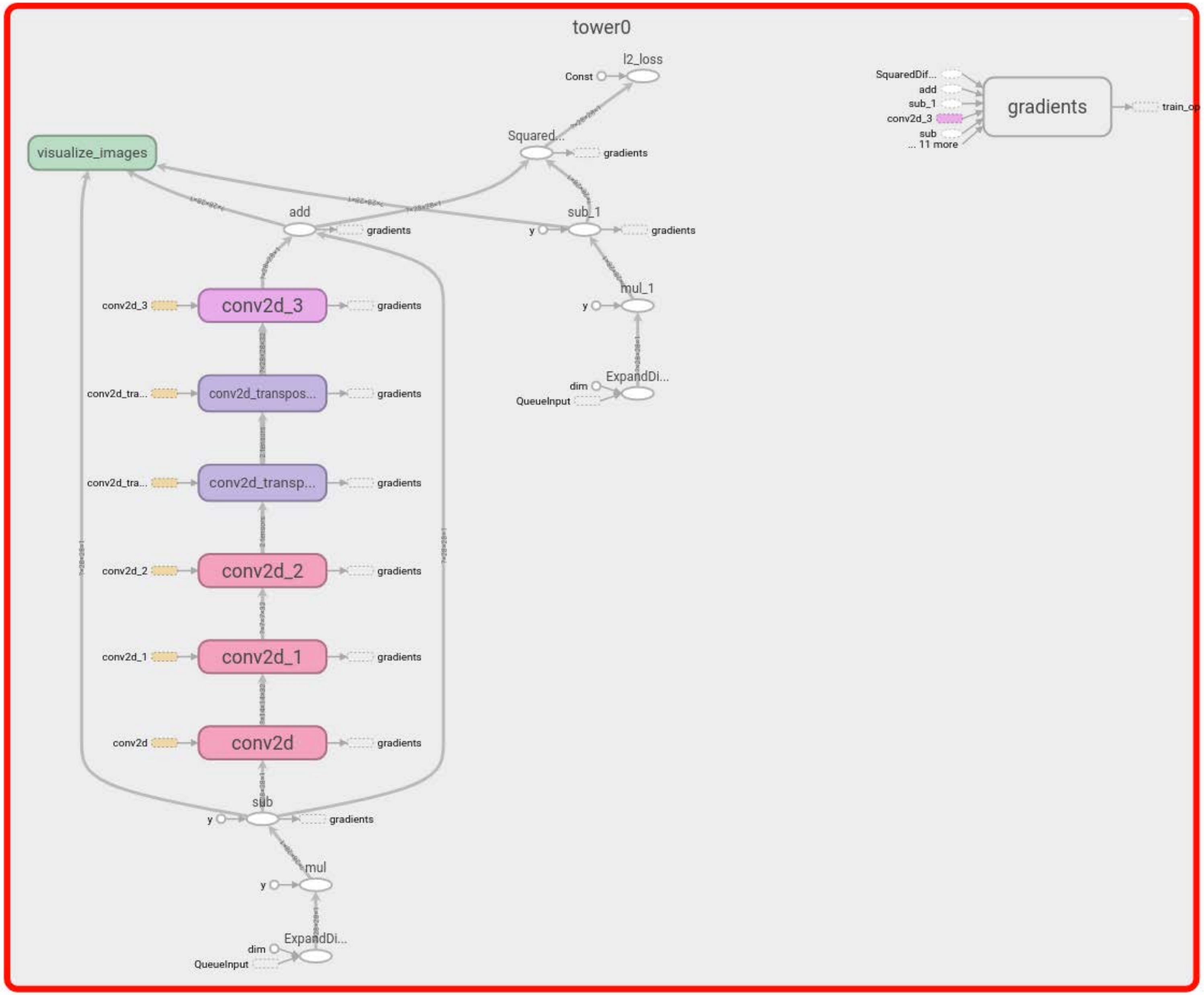
Terminal

```
→ ~/public_code/lectures/ml_in_cg_vision/Tutorial_01 :master✗ TF_CPP_MIN_LOG_LEVEL=3 python mnist_tensorpack_slide.py --gpu 0
[0516 10:38:03 @logger.py:74] Argv: mnist_tensorpack_slide.py --gpu 0
[0516 10:38:03 @interface.py:31] Automatically applying QueueInput on the DataFlow.
[0516 10:38:03 @input_source.py:193] Setting up the queue 'QueueInput/input_queue' for CPU prefetching ...
[0516 10:38:03 @training.py:101] Building graph for training tower 0 ...
[0516 10:38:04 @model_utils.py:63] Trainable Variables:
name            shape             dim
--------------  ---------------  ------
conv0/kernel:0  [3, 3, 1, 32]      288
conv0/bias:0    [32]                32
conv1/kernel:0  [3, 3, 32, 32]    9216
conv1/bias:0    [32]                32
conv2/kernel:0  [3, 3, 32, 32]    9216
conv2/bias:0    [32]                32
conv3/kernel:0  [3, 3, 32, 32]    9216
conv3/bias:0    [32]                32
fc0/kernel:0    [1568, 512]     802816
fc0/bias:0      [512]              512
fc1/kernel:0    [512, 10]         5120
fc1/bias:0      [10]                10
Total #vars=12, #params=836522, size=3.19MB
[0516 10:38:04 @base.py:208] Setup callbacks graph ...
[0516 10:38:04 @predict.py:41] Building predictor tower 'InferenceTower' on device /gpu:0 ...
[0516 10:38:04 @summary.py:38] Maintain moving average summary of 1 tensors in collection MOVING_SUMMARY_OPS.
[0516 10:38:04 @summary.py:75] Summarizing collection 'summaries' of size 2.
[0516 10:38:04 @base.py:226] Creating the session ...
[0516 10:38:09 @base.py:234] Initializing the session ...
[0516 10:38:09 @base.py:241] Graph Finalized.
[0516 10:38:09 @concurrency.py:37] Starting EnqueueThread QueueInput/input_queue ...
[0516 10:38:09 @inference_runner.py:99] InferenceRunner will eval 40 iterations
[0516 10:38:10 @base.py:261] Start Epoch 1 ...
100%|########################################################################################################################|468/468[00:07<00:00,63.38it/s]
[0516 10:38:17 @base.py:271] Epoch 1 (global_step 468) finished, time:7.38 seconds.
[0516 10:38:17 @saver.py:84] Model saved to train_log/mnist_tensorpack_slide/model-468.
100%|#########################################################################################################################|40/40[00:00<00:00,100.20it/s]
[0516 10:38:17 @monitor.py:433] QueueInput/queue_size: 50
[0516 10:38:17 @monitor.py:433] accuracy: 0.97561
[0516 10:38:17 @monitor.py:433] validation_accuracy: 0.98643
[0516 10:38:17 @monitor.py:433] validation_cross_entropy_loss: 0.042295
[0516 10:38:17 @base.py:261] Start Epoch 2 ...
100%|#######################################################################################################################|468/468[00:04<00:00,109.56it/s]
[0516 10:38:22 @base.py:271] Epoch 2 (global_step 936) finished, time:4.27 seconds.
```

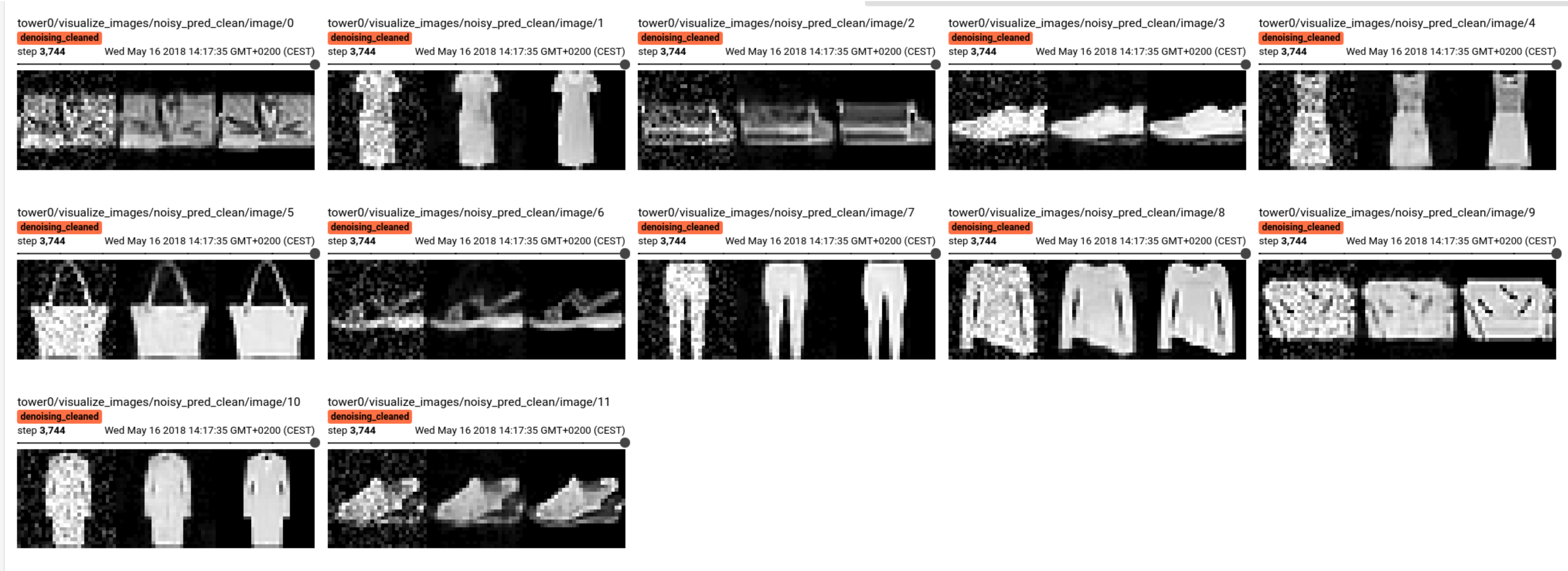
Tensorboard - Overview



Tensorboard



Denoising Task



Summary

- Convolutional Neural Networks – Toolbox for assembling nets for images
  - Conv Layer
  - Dropout
  - Max Pool
  - Activation Function
  - Batchnorm
  - ….

References

- G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors.
- A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. www.imagenet.org/challenges. 2010.
- Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012
- Karen Simonyan, Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015
- Matthew D. Zeiler and Rob Fergus, Visualizing and Understanding Convolutional Networks, ECCV 2014
- S. Ioffe & C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML 2015
- Olaf Ronneberger, Philipp Fischer, Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015
- Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, PAMI 2017