Convolutional Neural Networks

Ahmed Hosny

HARVARD MEDICAL SCHOOL | BRIGHAM AND WOMEN'S HOSPITAL | DANA-FARBER CANCER INSTITUTE

SAM Joint Imaging-Therapy Scientific Symposium (Certificate Series Session 3)
Convolutional Neural Nets - Wednesday, 8/1/2018 1:45 PM - 3:45 PM

Why do we need convolutions?

CNN specifics

CNN flavors

Resources

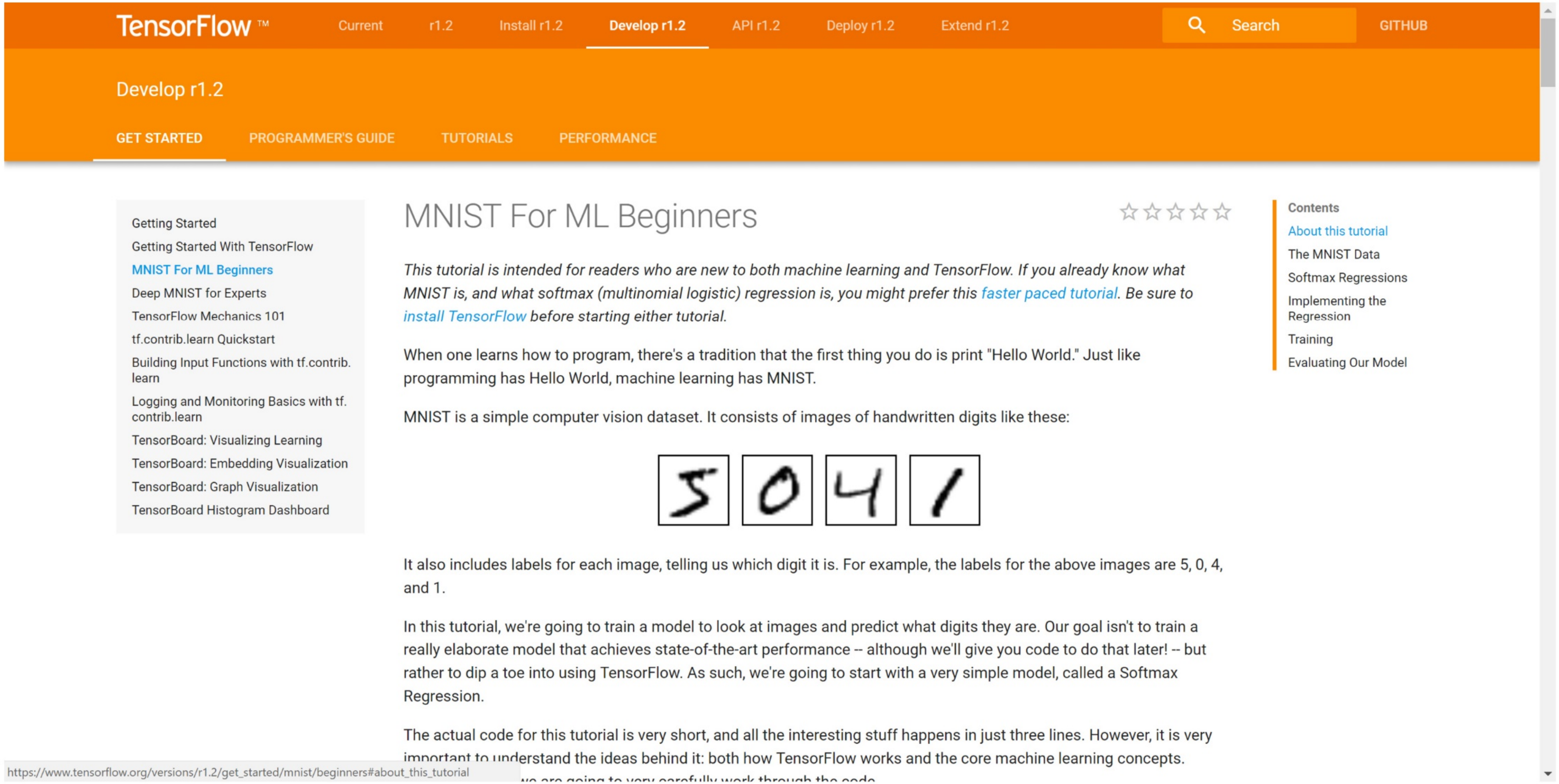
Why do we need convolutions?

CNN specifics

CNN flavors

Resources

TensorFlow - MNIST For Beginners



TensorFlow ™ Current r1.2 Install r1.2 Develop r1.2 API r1.2 Deploy r1.2 Extend r1.2 Search GITHUB

Develop r1.2

GET STARTED PROGRAMMER'S GUIDE TUTORIALS PERFORMANCE

Getting Started
Getting Started With TensorFlow
MNIST For ML Beginners
Deep MNIST for Experts
TensorFlow Mechanics 101
tf.contrib.learn Quickstart
Building Input Functions with tf.contrib.learn
Logging and Monitoring Basics with tf.contrib.learn
TensorBoard: Visualizing Learning
TensorBoard: Embedding Visualization
TensorBoard: Graph Visualization
TensorBoard Histogram Dashboard

MNIST For ML Beginners

This tutorial is intended for readers who are new to both machine learning and TensorFlow. If you already know what MNIST is, and what softmax (multinomial logistic) regression is, you might prefer this faster paced tutorial. Be sure to install TensorFlow before starting either tutorial.

When one learns how to program, there's a tradition that the first thing you do is print "Hello World." Just like programming has Hello World, machine learning has MNIST.

MNIST is a simple computer vision dataset. It consists of images of handwritten digits like these:

It also includes labels for each image, telling us which digit it is. For example, the labels for the above images are 5, 0, 4, and 1.

In this tutorial, we're going to train a model to look at images and predict what digits they are. Our goal isn't to train a really elaborate model that achieves state-of-the-art performance -- although we'll give you code to do that later! -- but rather to dip a toe into using TensorFlow. As such, we're going to start with a very simple model, called a Softmax Regression.

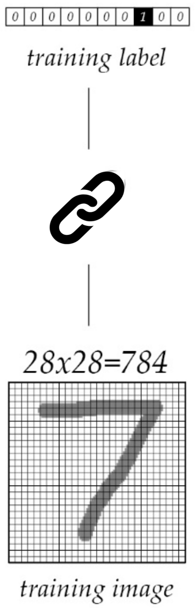
The actual code for this tutorial is very short, and all the interesting stuff happens in just three lines. However, it is very important to understand the ideas behind it: both how TensorFlow works and the core machine learning concepts.

Contents
About this tutorial
The MNIST Data
Softmax Regressions
Implementing the Regression
Training
Evaluating Our Model

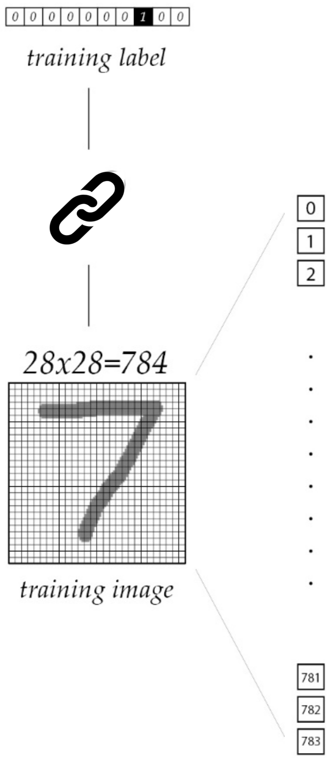
https://www.tensorflow.org/versions/r1.2/get_started/mnist/beginners#about_this_tutorial

https://www.tensorflow.org/versions/r1.2/get_started/mnist/beginners

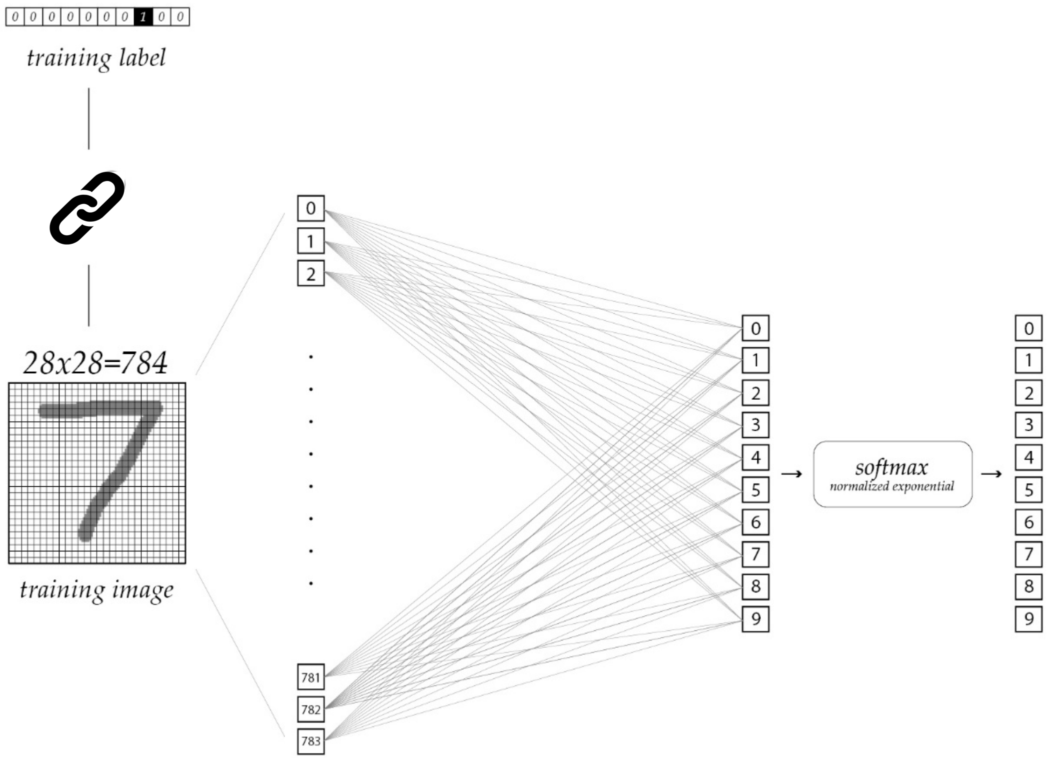
TensorFlow - MNIST For Beginners



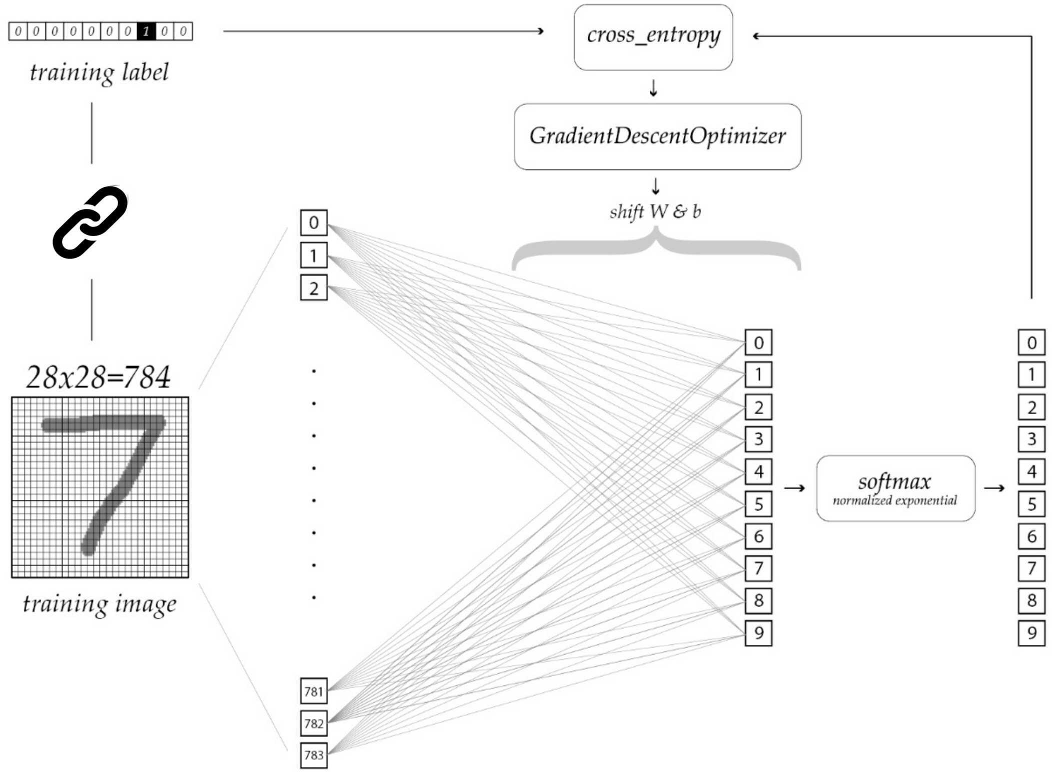
TensorFlow - MNIST For Beginners



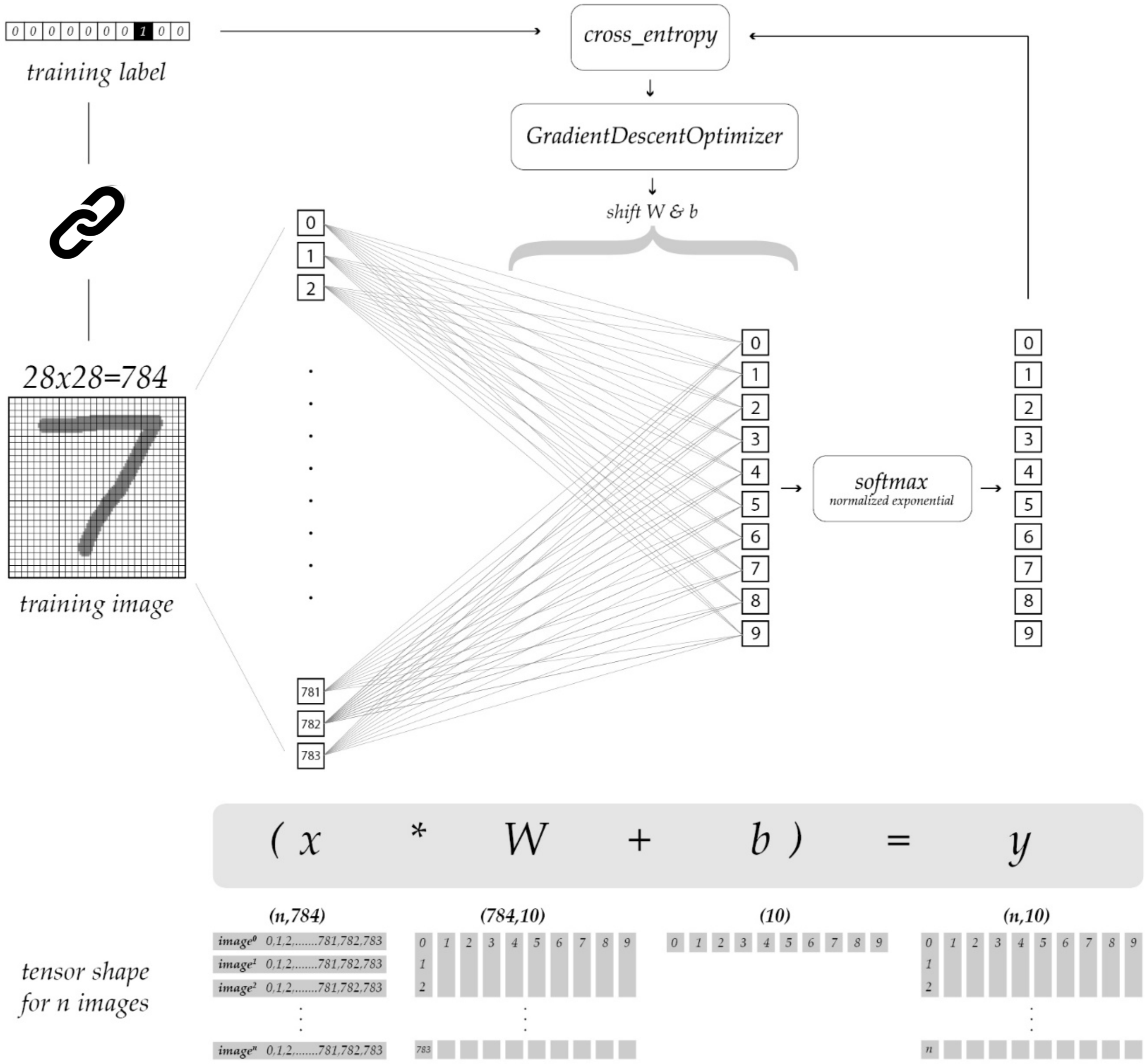
TensorFlow - MNIST For Beginners



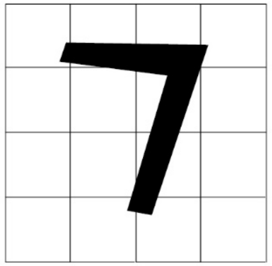
TensorFlow - MNIST For Beginners



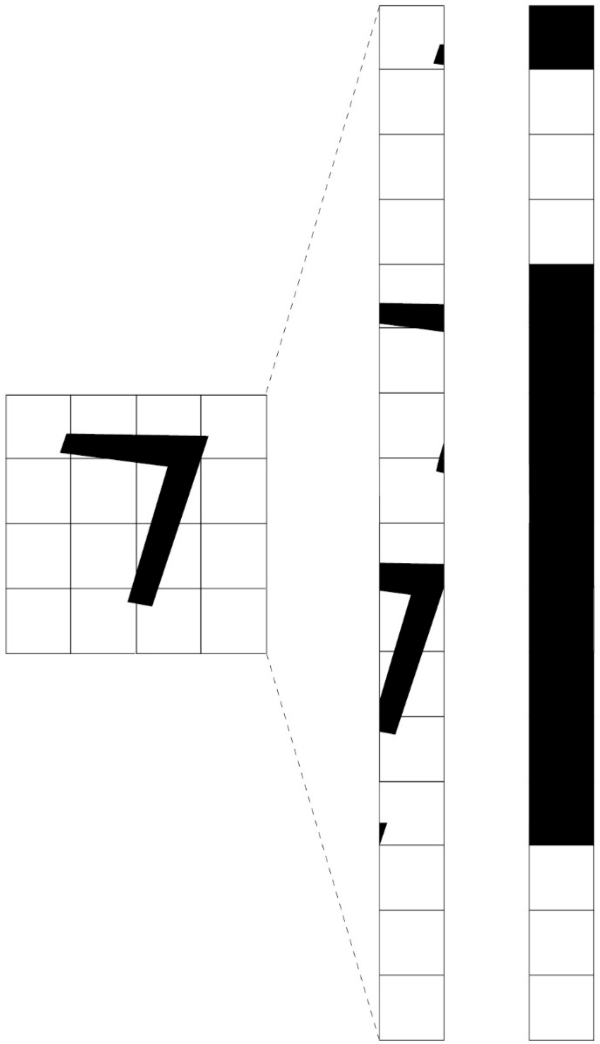
TensorFlow - MNIST For Beginners



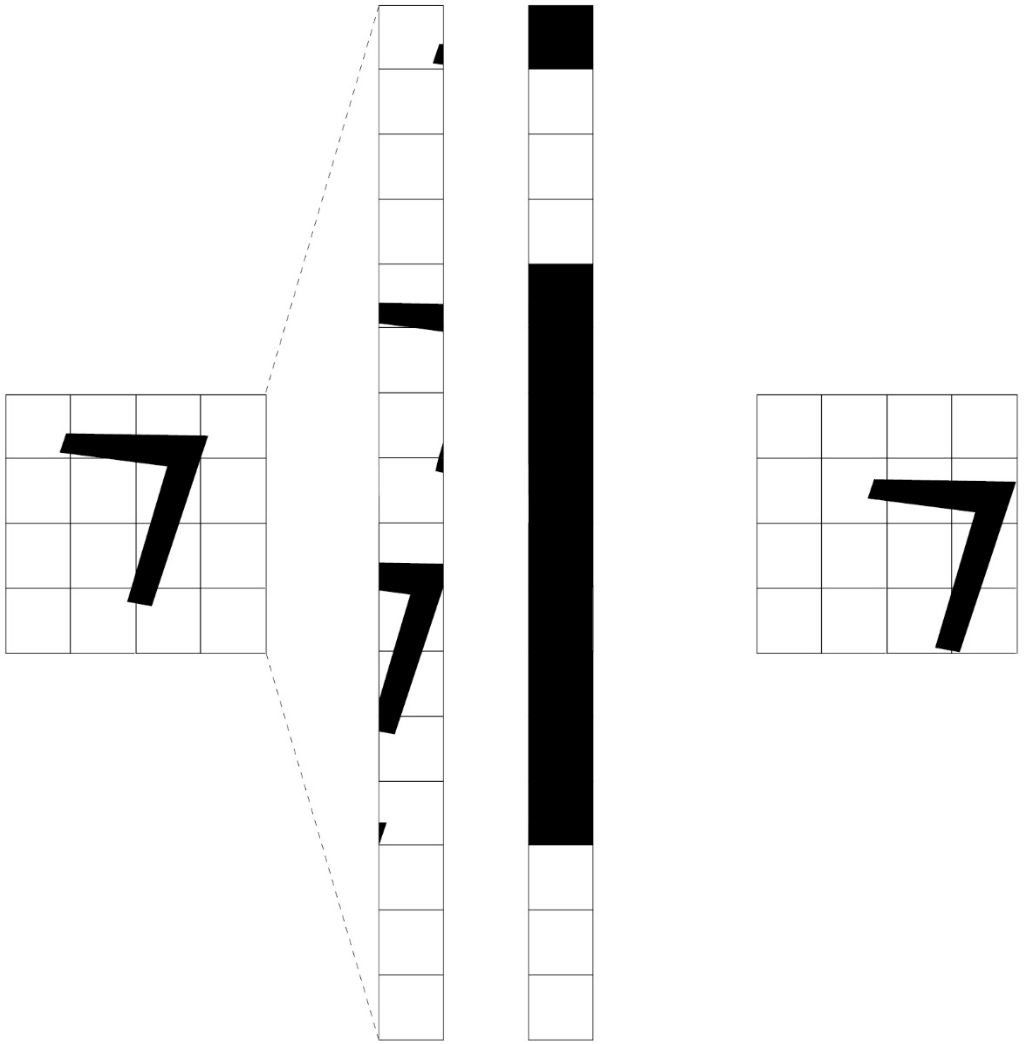
Translation Variance



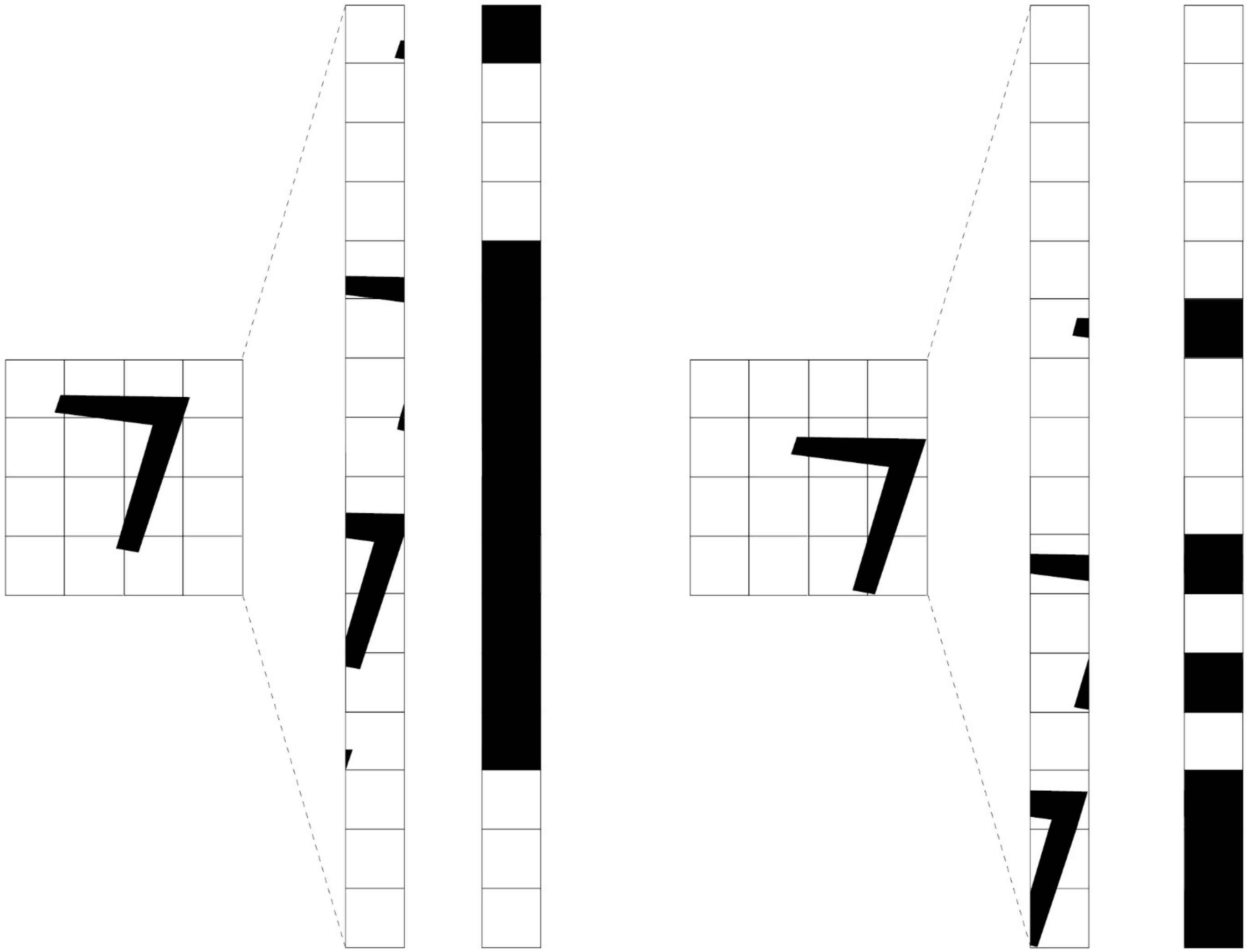
Translation Variance



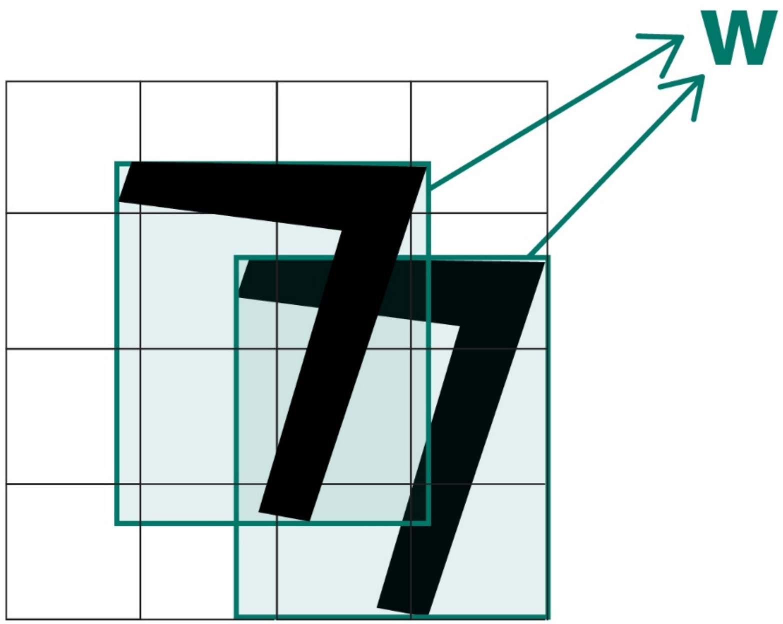
Translation Variance



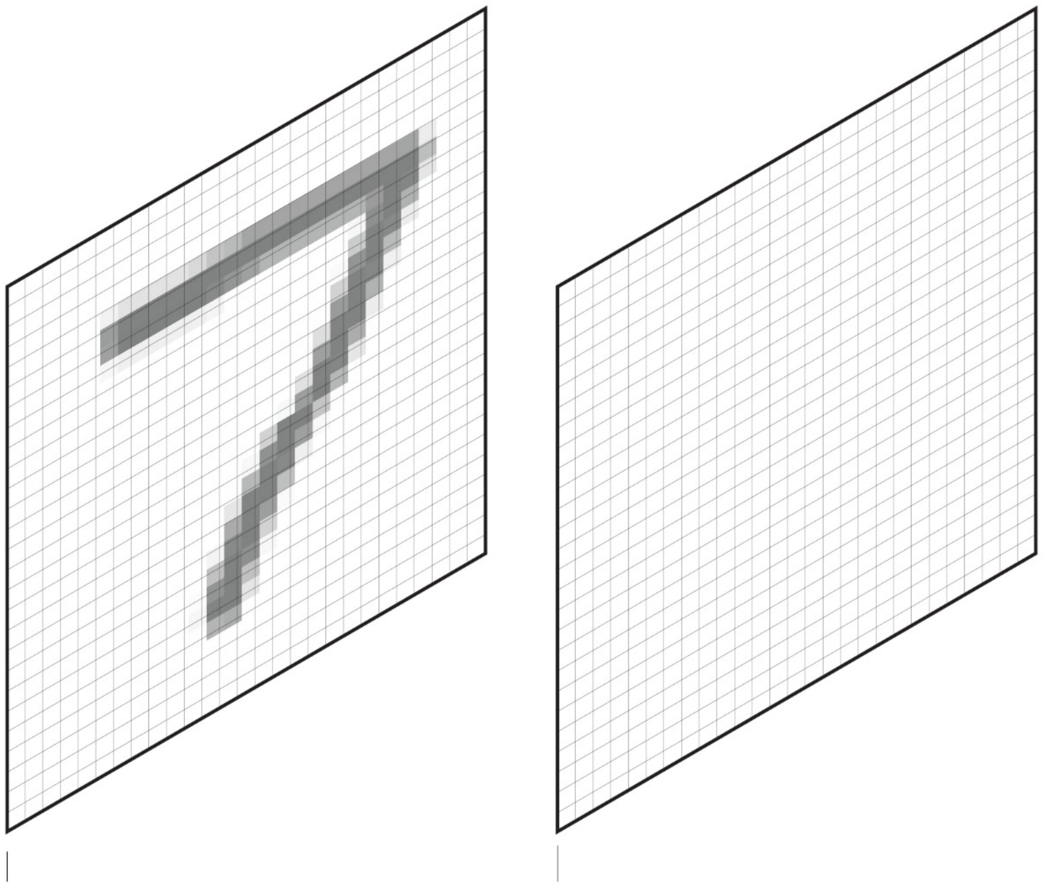
Translation Variance



Weight Sharing



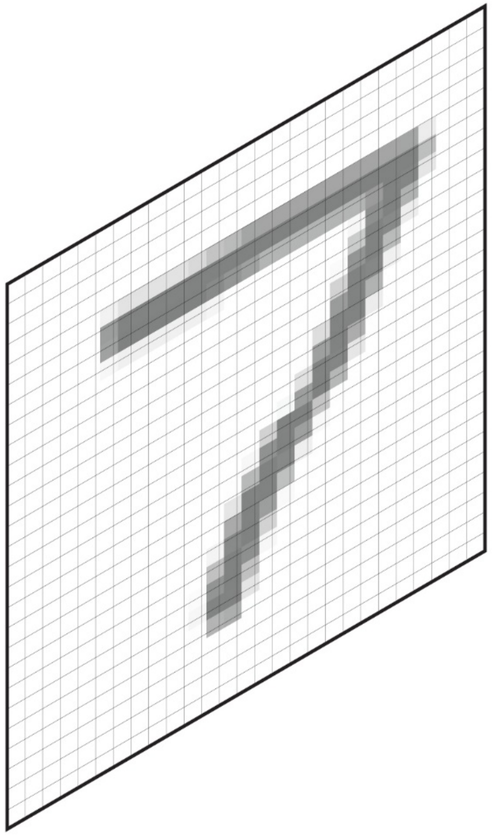
Convolutions



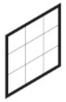
input
28x28

convolution
filter/kernel

Convolutions

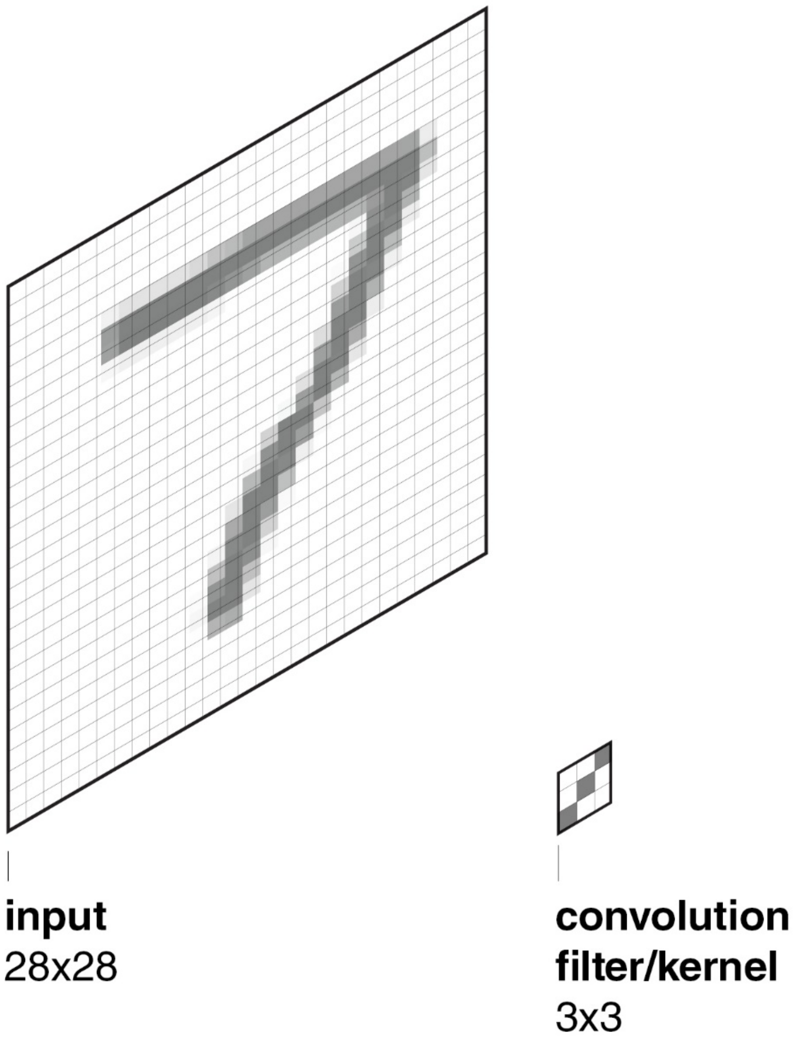


input
28x28

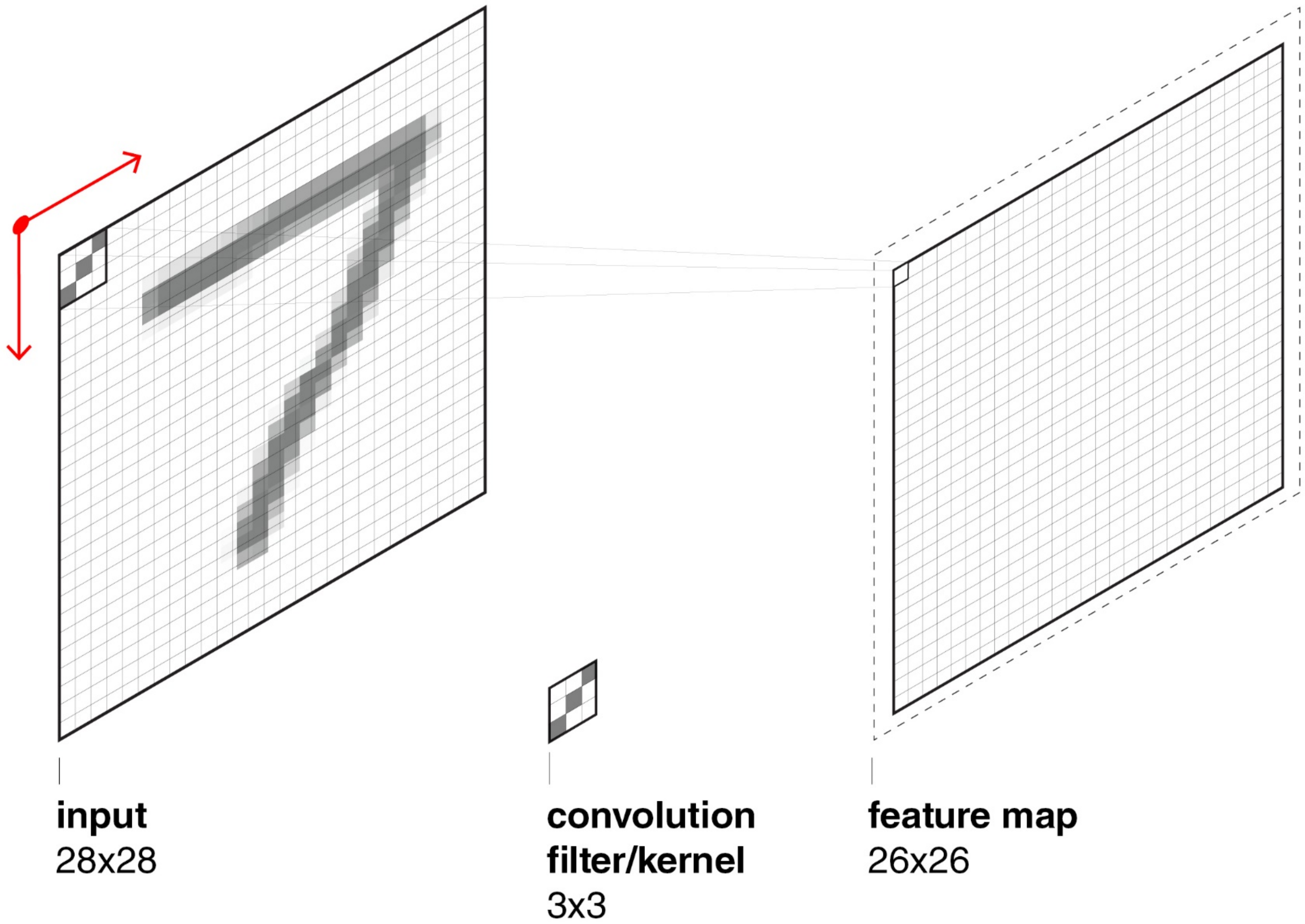


convolution filter/kernel
3x3

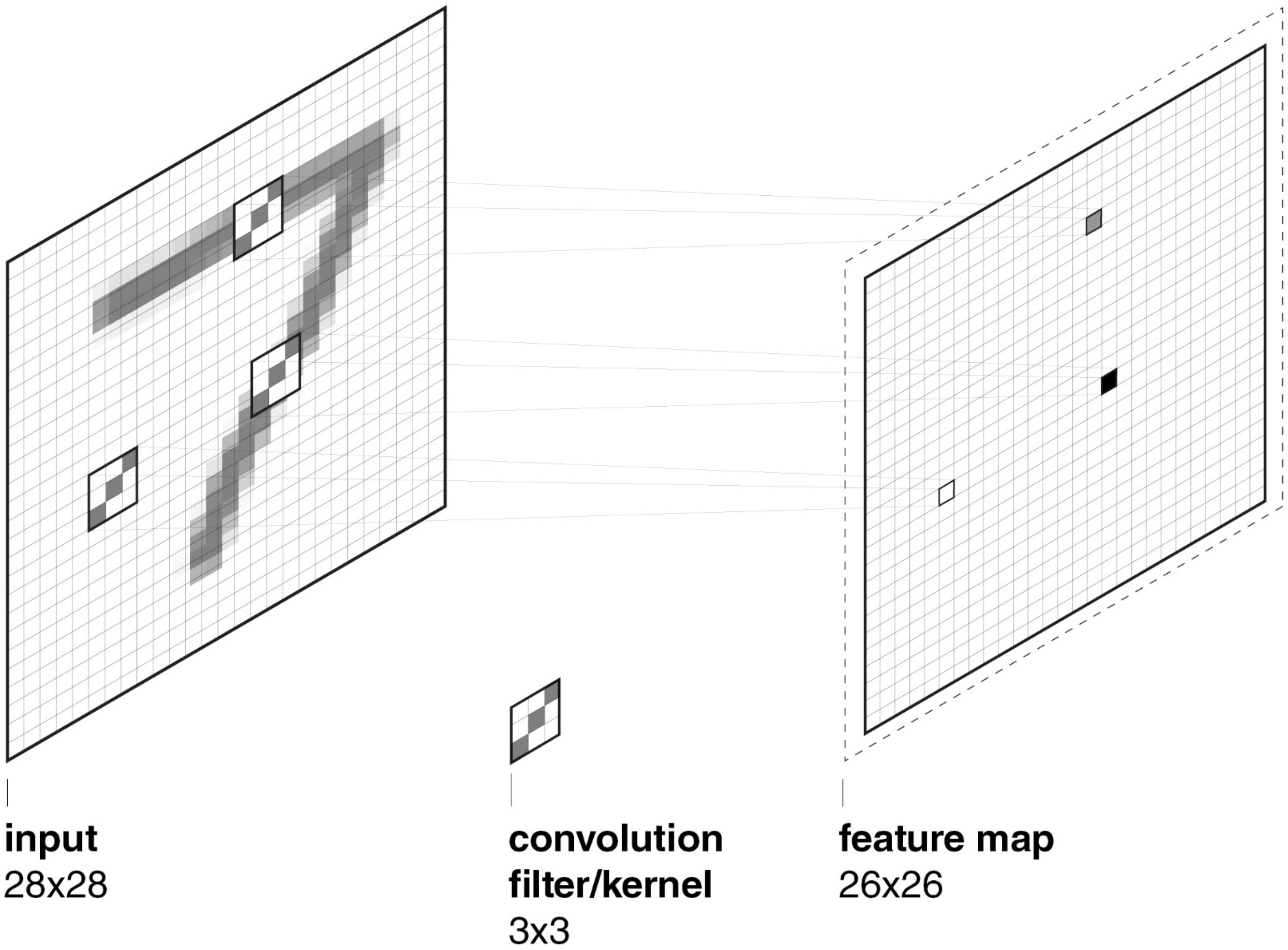
Convolutions



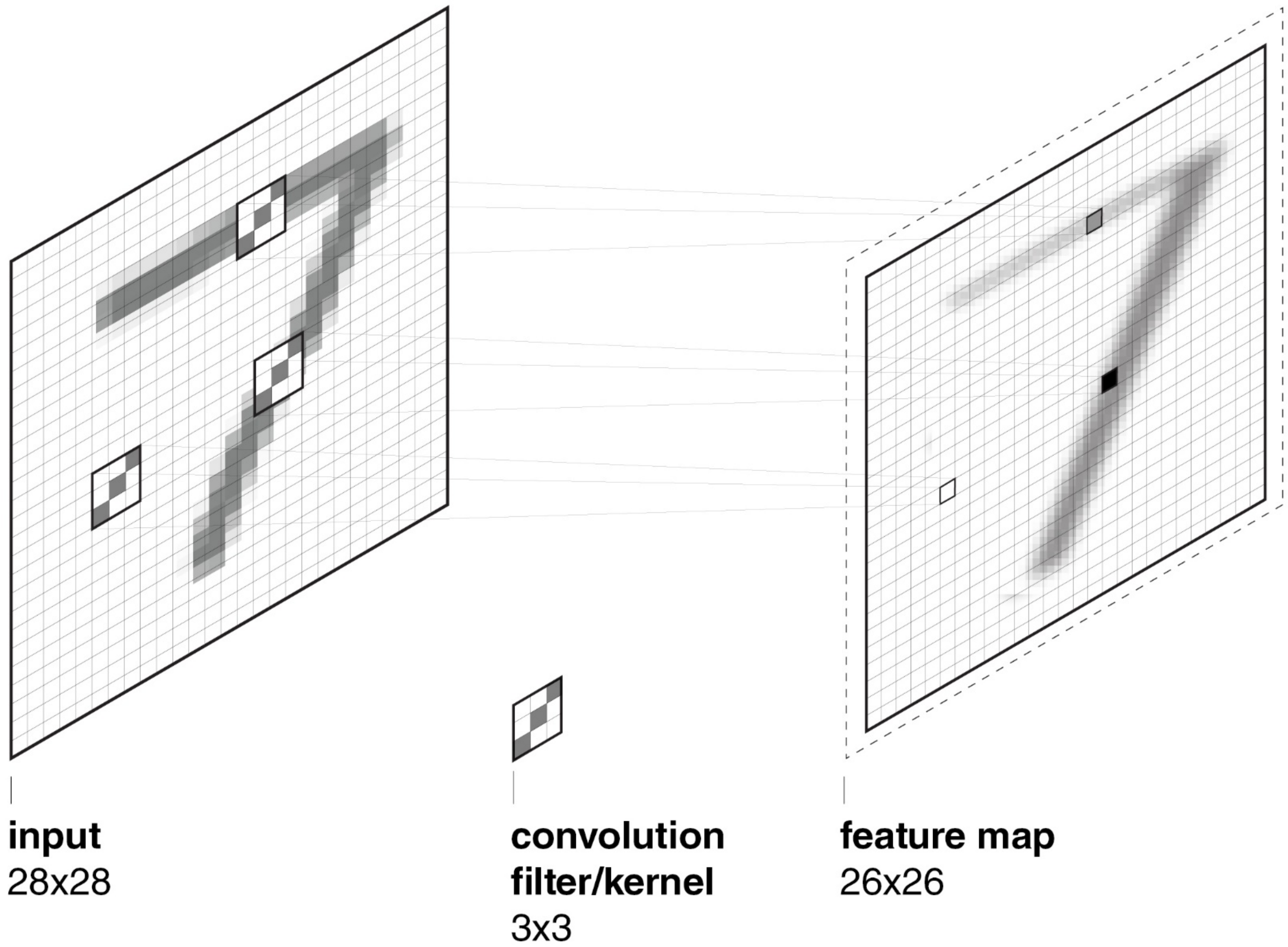
Convolutions



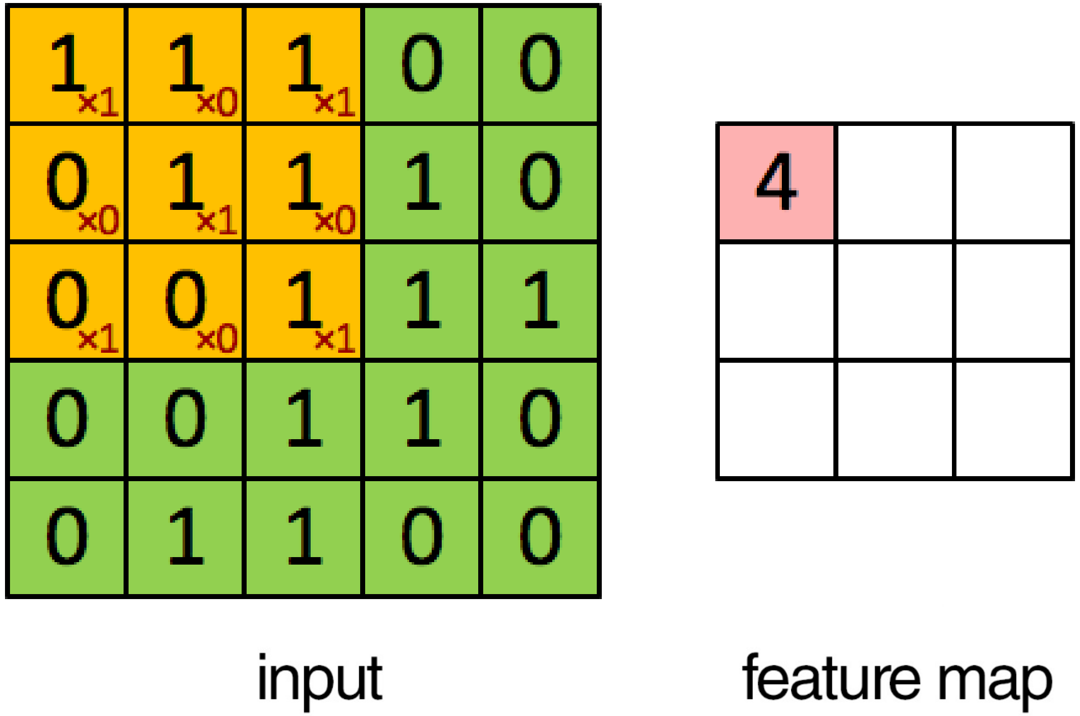
Convolutions



Convolutions

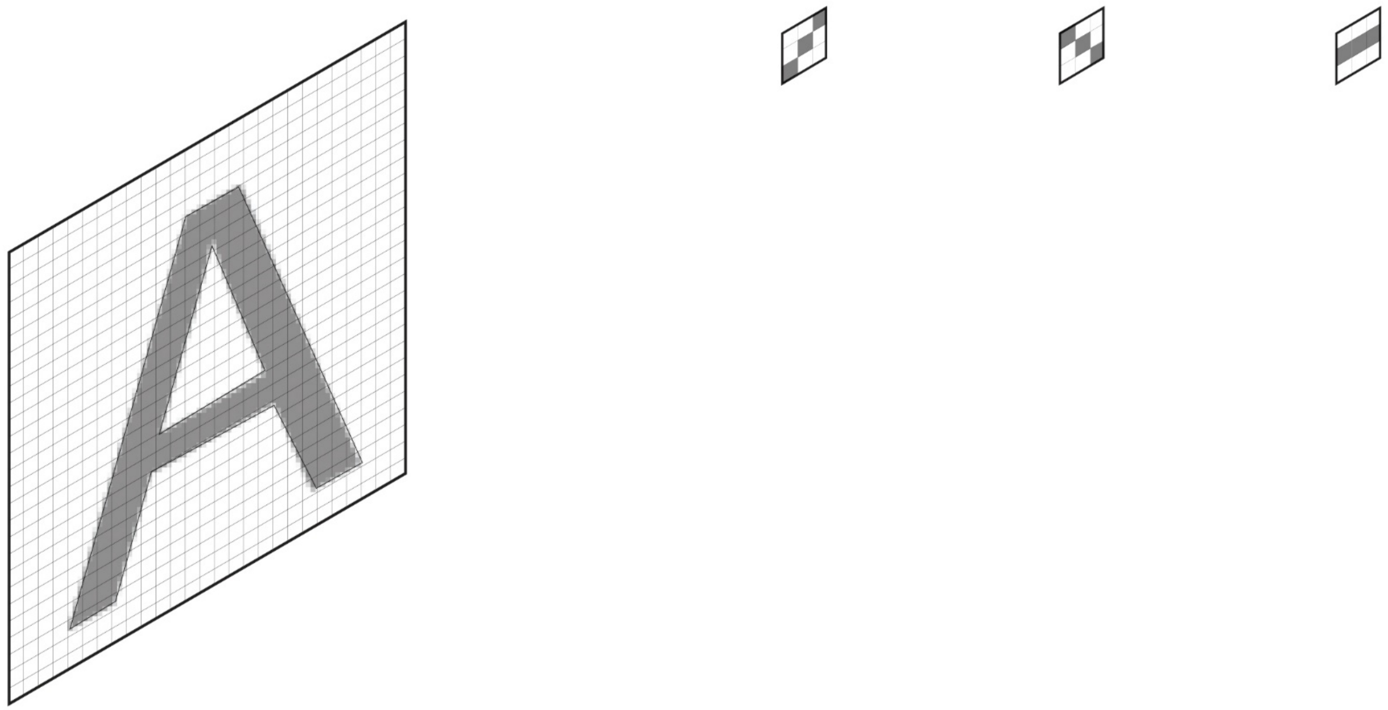


Convolutions

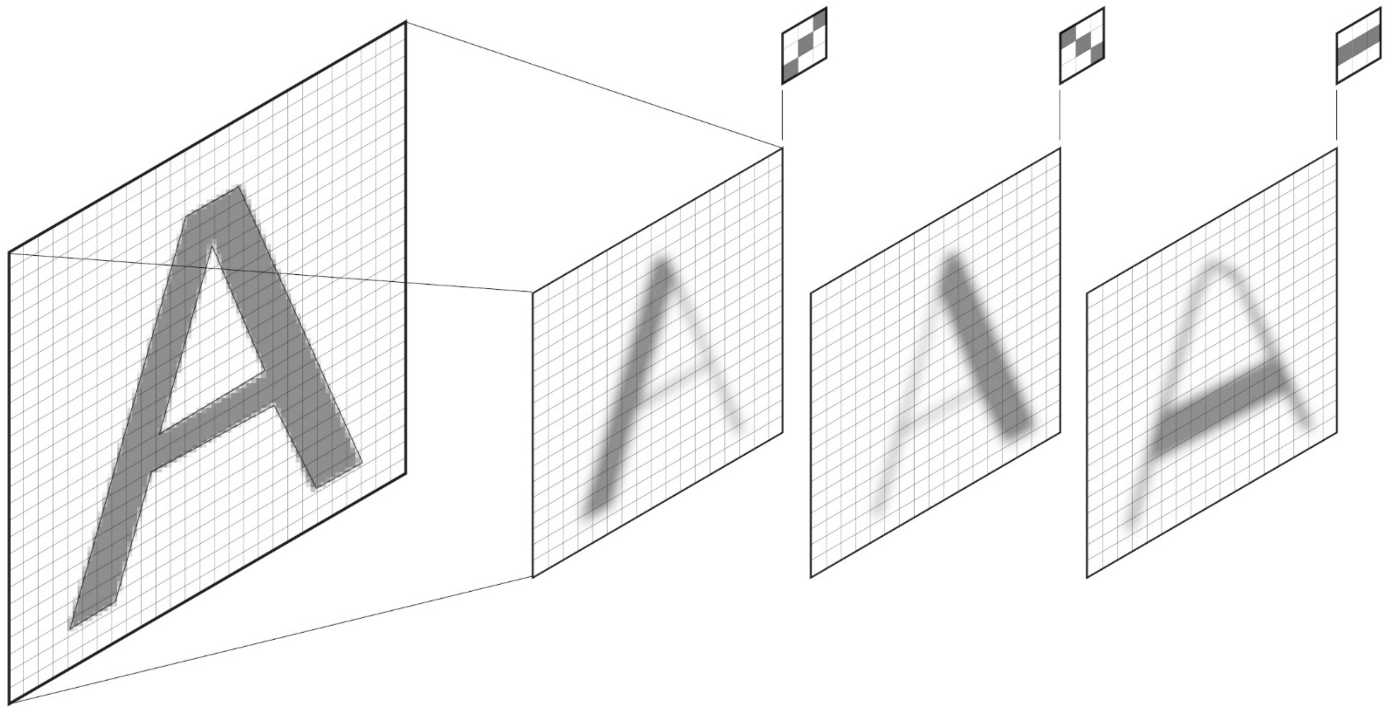


https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

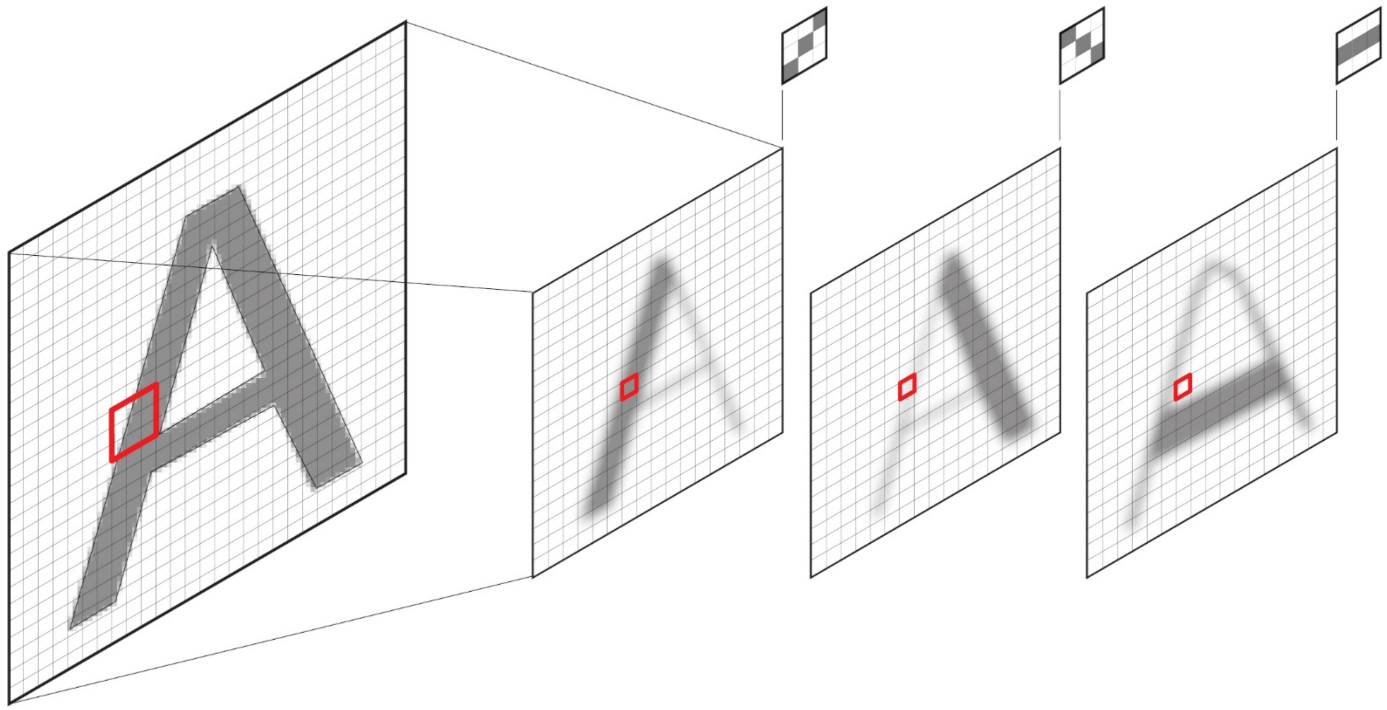
Convolutions



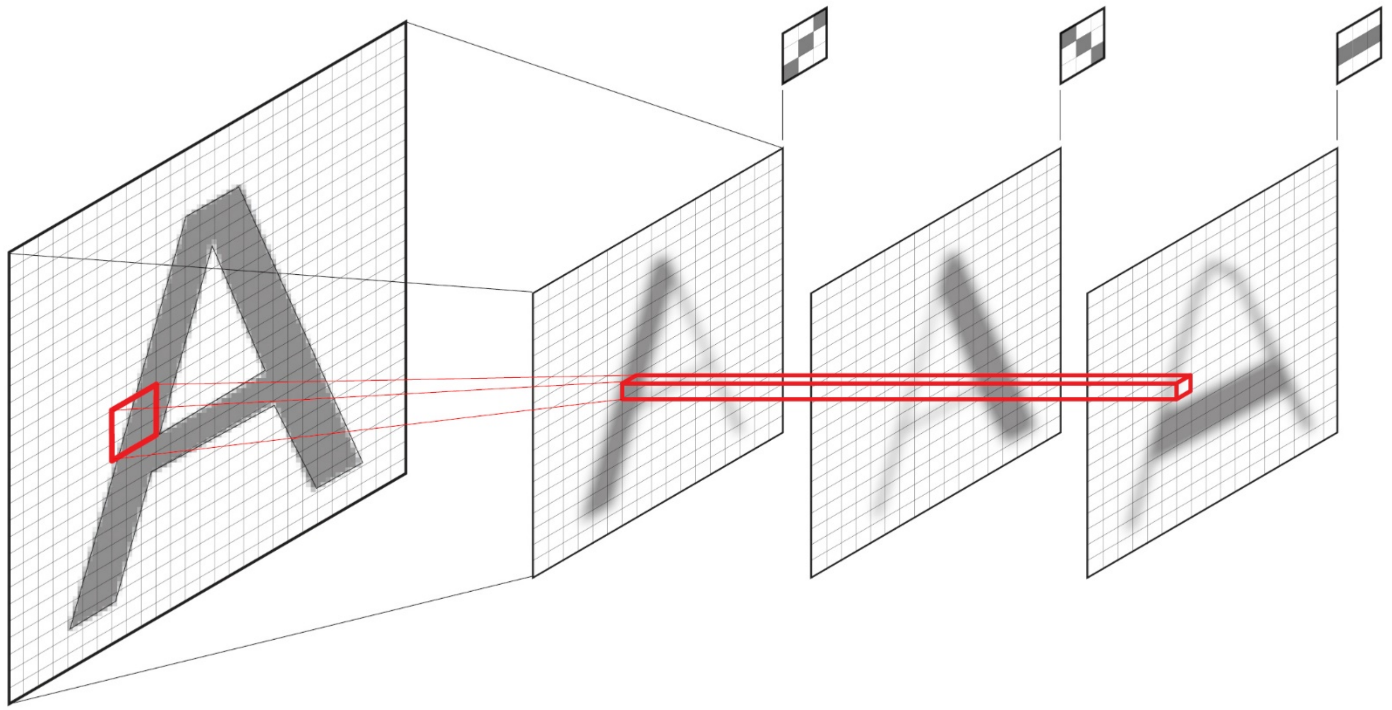
Convolutions



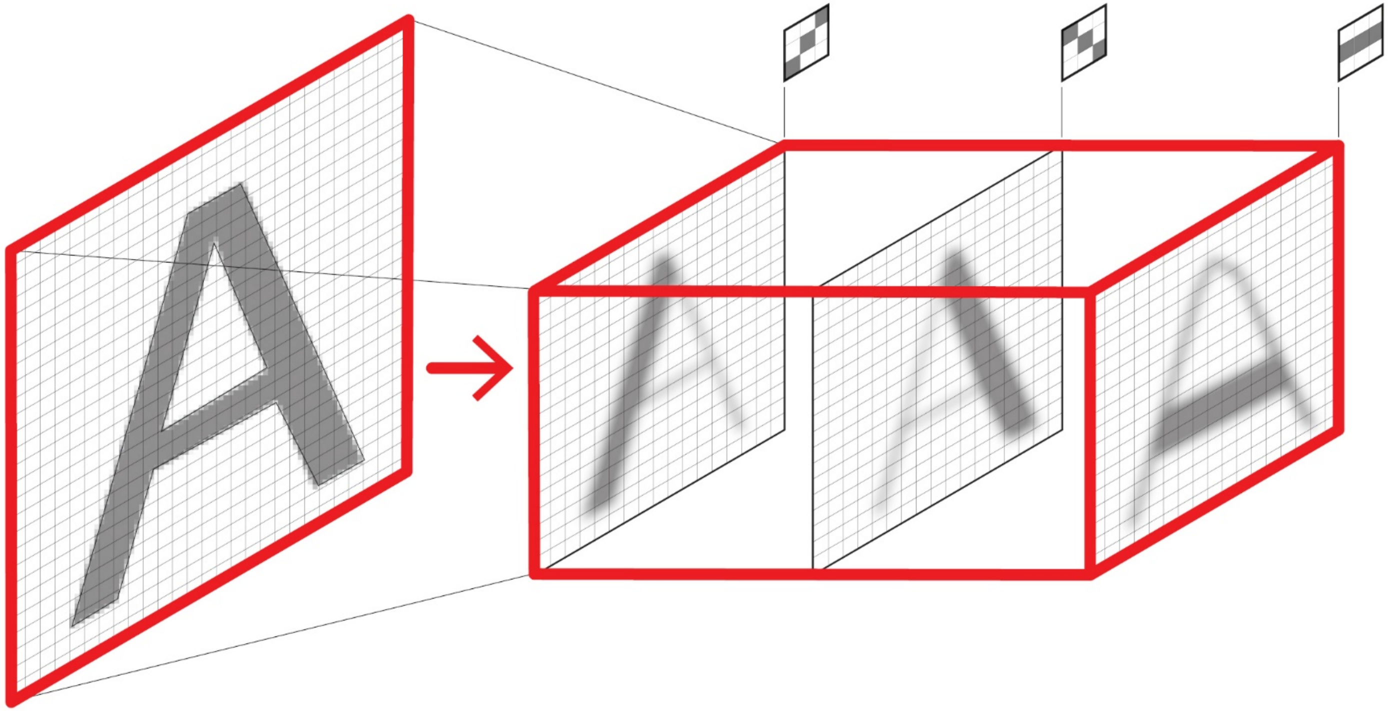
Convolutions



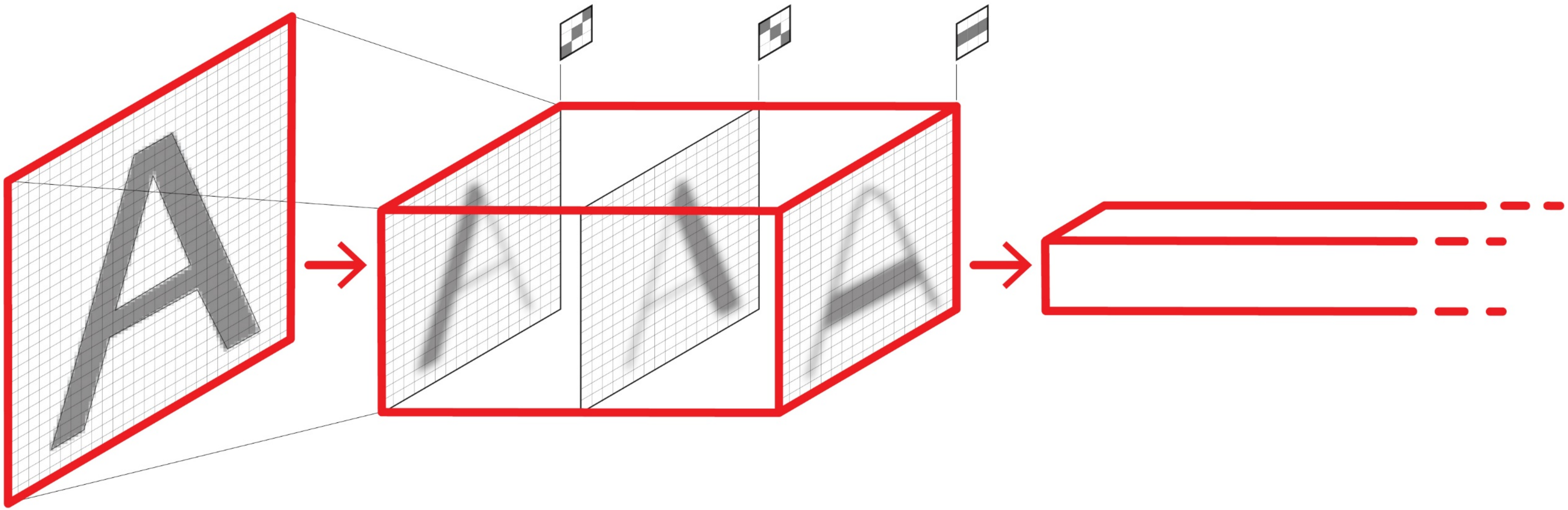
Convolutions



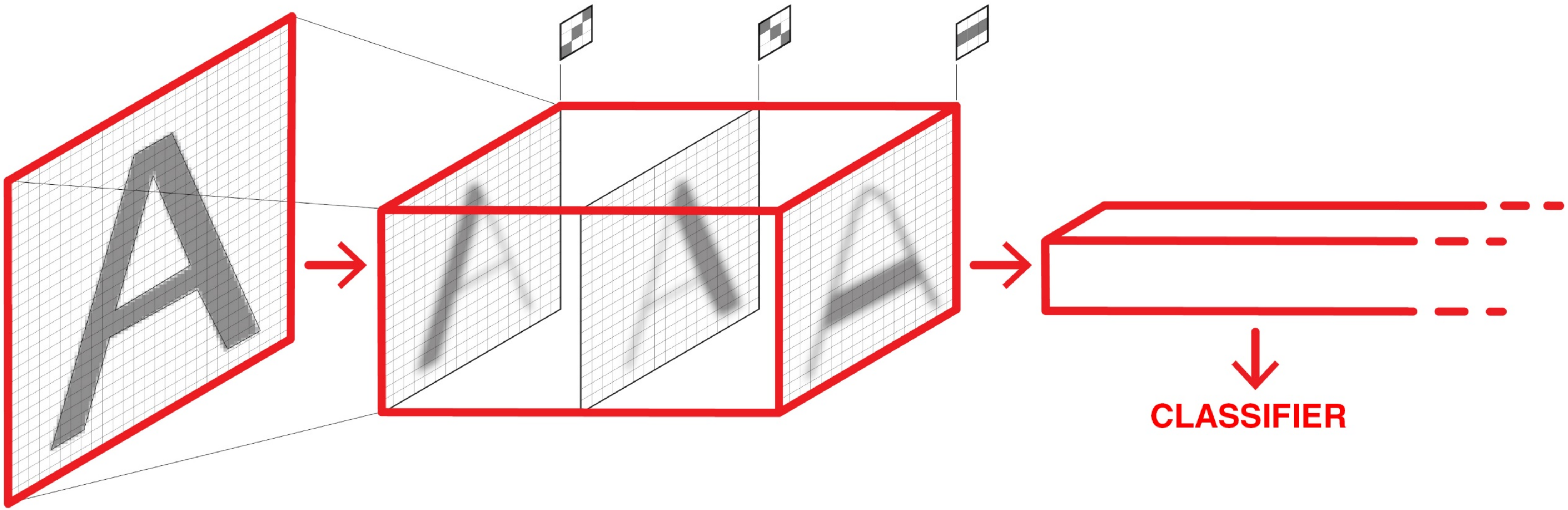
Convolutions



Convolutions



Convolutions



Why do we need convolutions?

CNN specifics

CNN flavors

Resources

Neocognitron

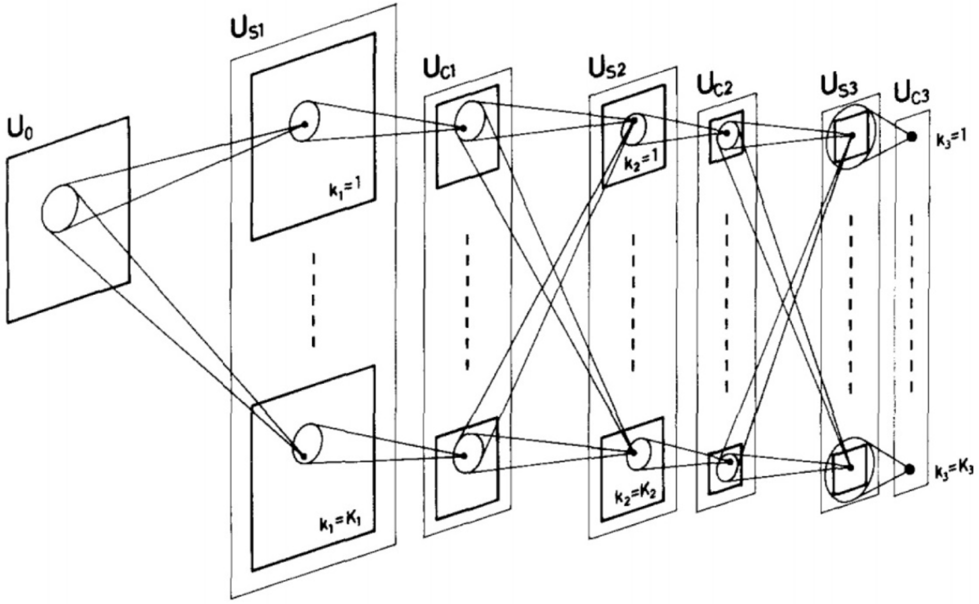


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

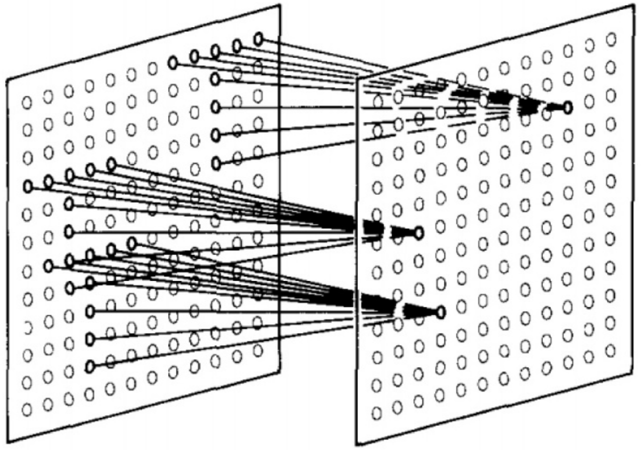


Fig. 3. Illustration showing the input interconnections to the cells within a single cell-plane

Kunihiko Fukushima

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Biological Cybernetics - 1980

LeNet-5

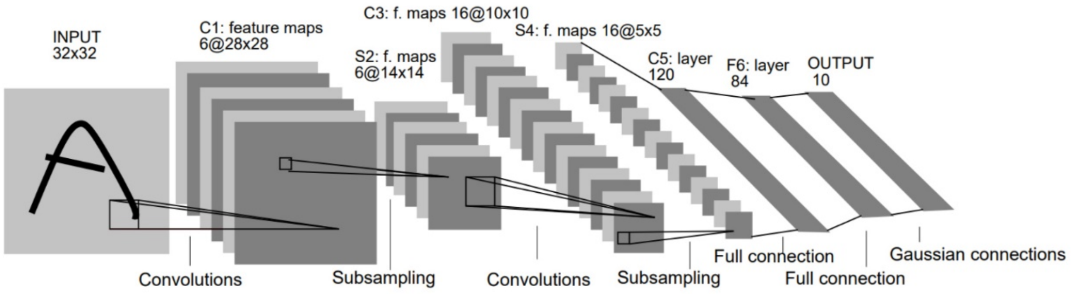


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

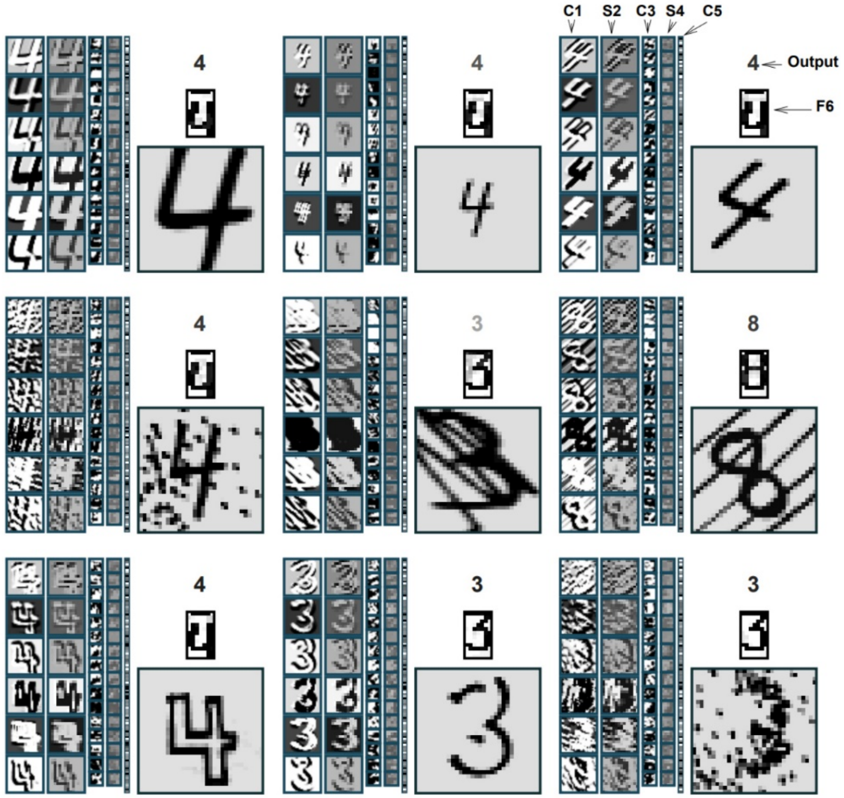


Fig. 4. Examples of unusual, distorted, and noisy characters correctly recognized by LeNet-5. The grey-level of the output label represents the penalty (lighter for higher penalties).

Yann LeCun, Patrick Haffner, Léon Bottou & Yoshua Bengio

Object Recognition with Gradient Based Learning

Shape, Contour and Grouping in Computer Vision - 1999

AlexNet

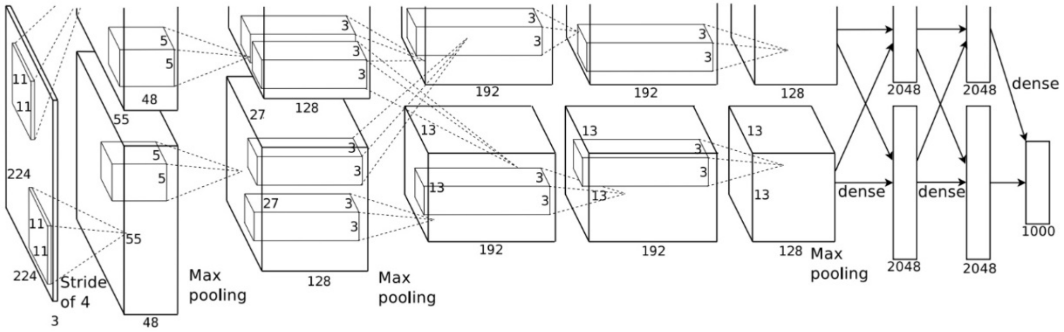


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton

ImageNet Classification with Deep Convolutional Neural Networks

Advances in Neural Information Processing Systems - 2012

AlexNet @ ImageNet

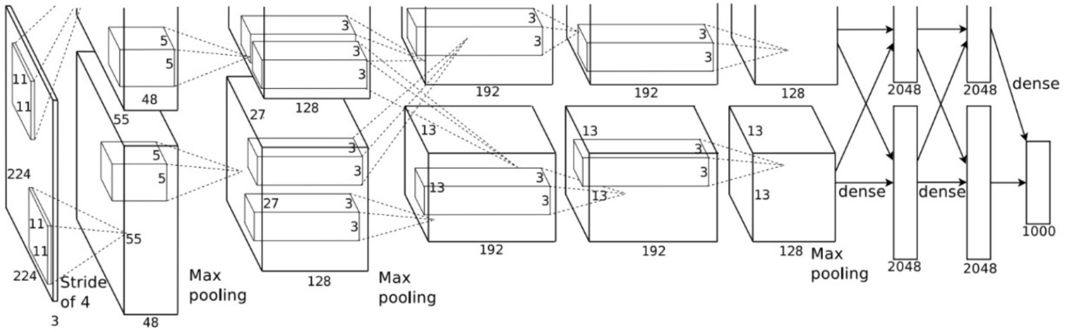
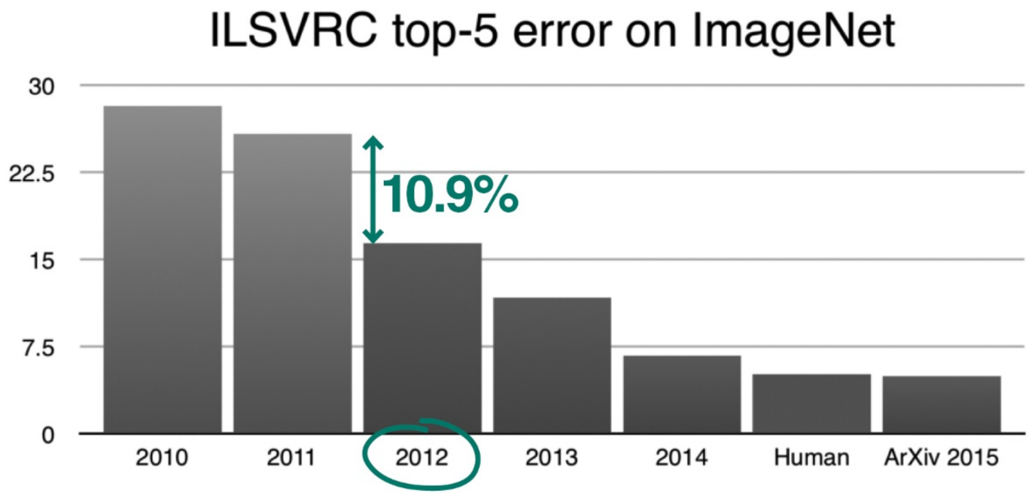
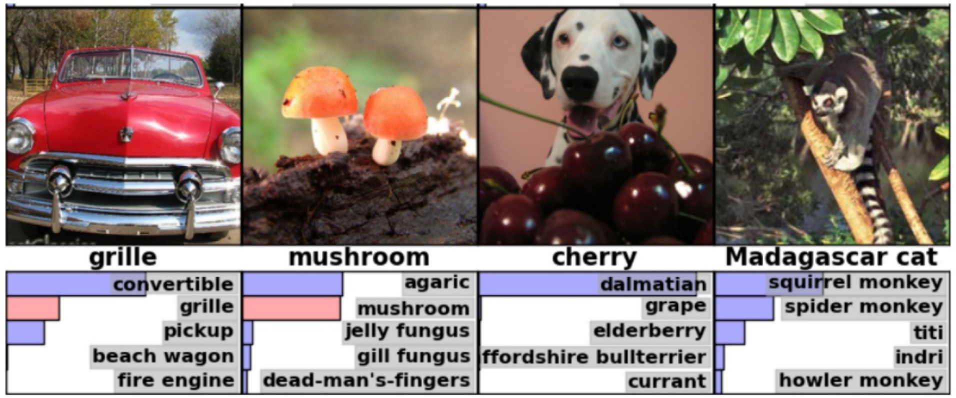


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.





https://devblogs.nvidia.com/mocha-jl-deep-learning-julia/

Hyperparameters

# of filters

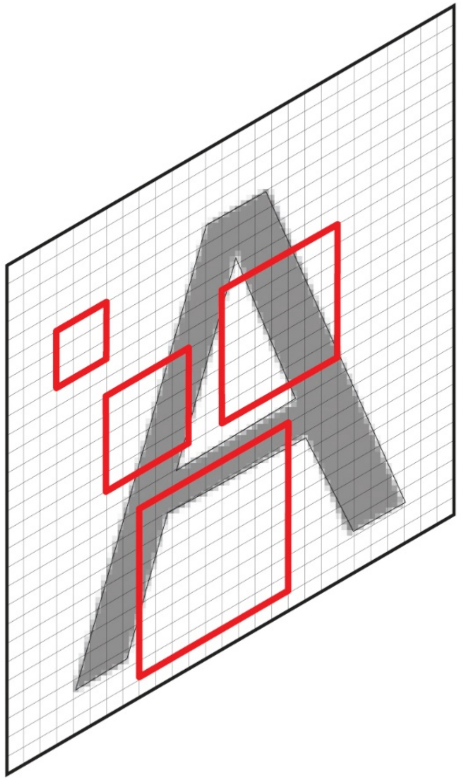
32, 64, 128 ...

https://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

Hyperparameters

filter size

11x11, 5x5, 3x3 ...



Hyperparameters

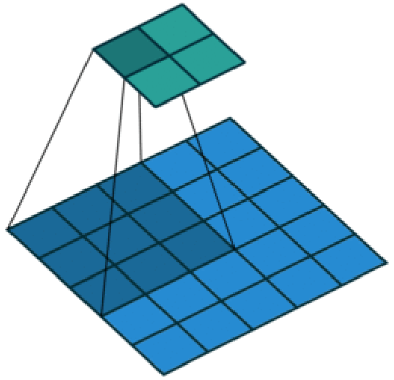
padding

pad the input image?

stride

# of pixels to shift the filter

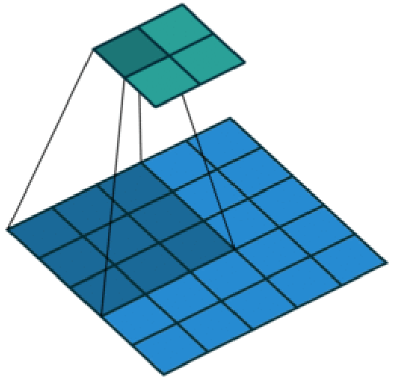
Hyperparameters



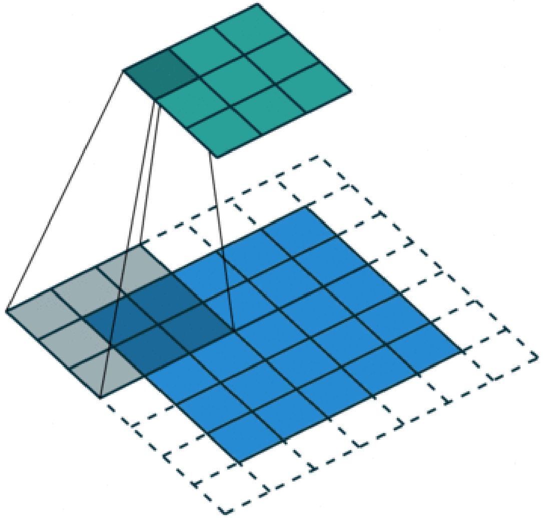
padding **no**
stride **2**

https://github.com/vdumoulin/conv_arithmetic

Hyperparameters



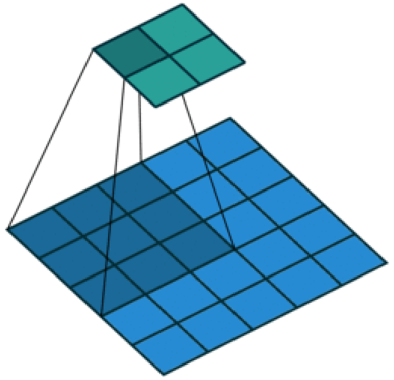
padding **no**
stride **2**



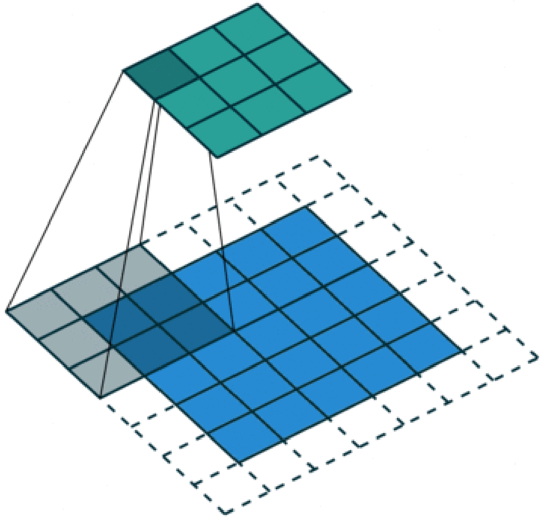
padding **yes**
stride **2**

https://github.com/vdumoulin/conv_arithmetic

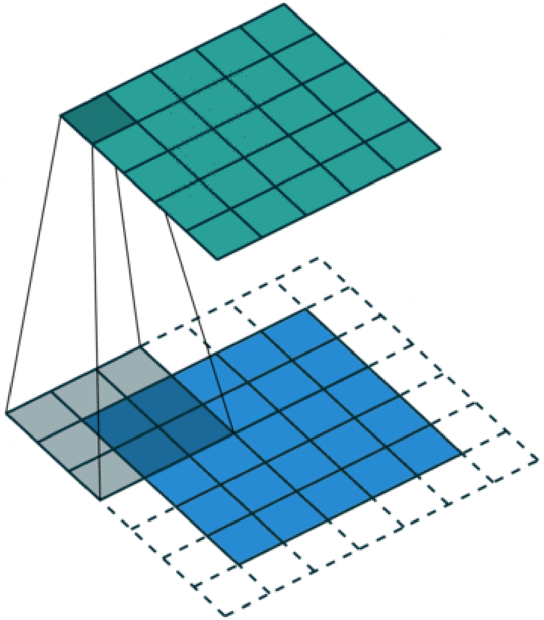
Hyperparameters



padding **no**
stride **2**



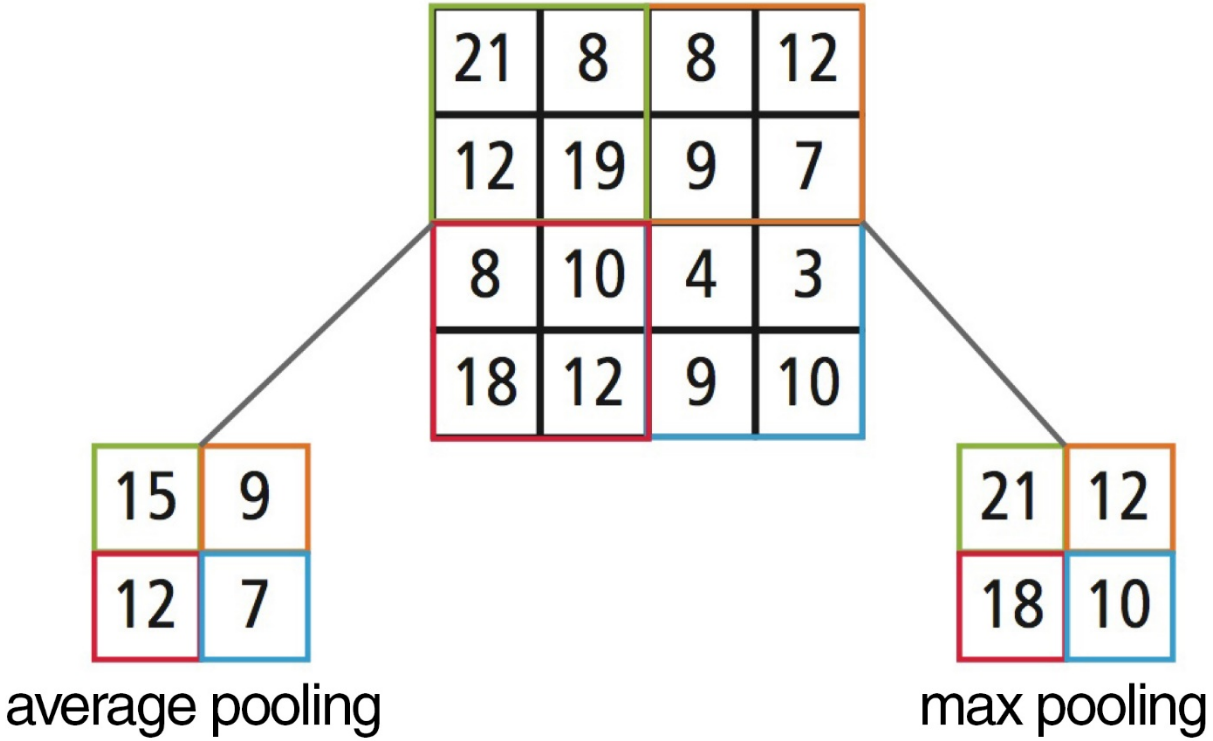
padding **yes**
stride **2**



padding **yes**
stride **1**

https://github.com/vdumoulin/conv_arithmetic

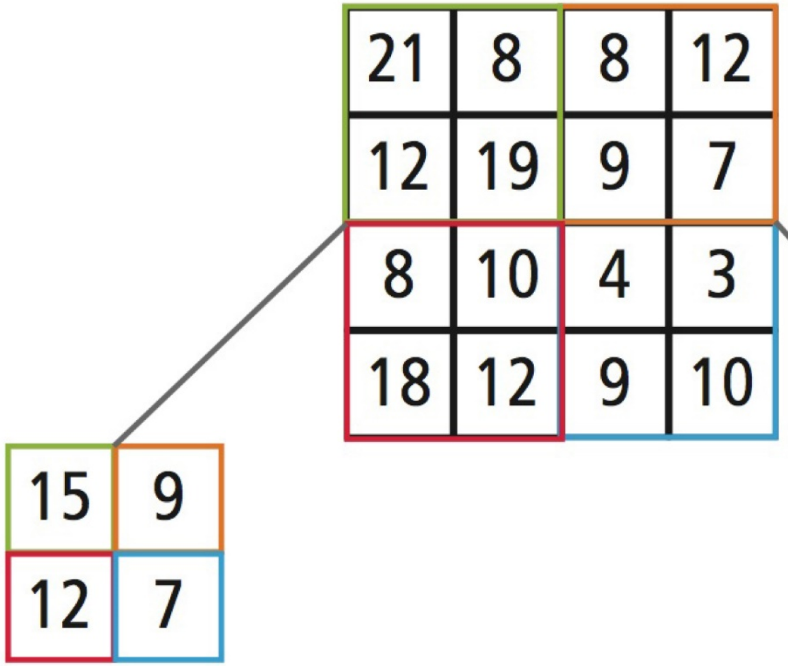
Pooling



https://www.embedded-vision.com/platinum-members/cadence/embedded-vision-training/documents/pages/neuralnetworksimagerecognition#3

Pooling

pooling size **2x2**
pooling stride **2**

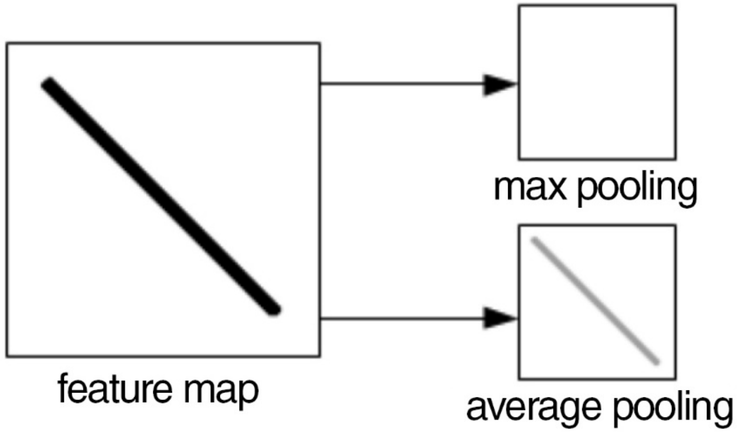


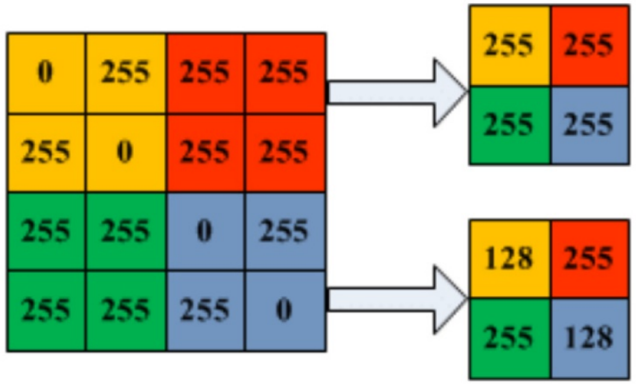
average pooling

max pooling

https://www.embedded-vision.com/platinum-members/cadence/embedded-vision-training/documents/pages/neuralnetworksimagerecognition#3

Pooling



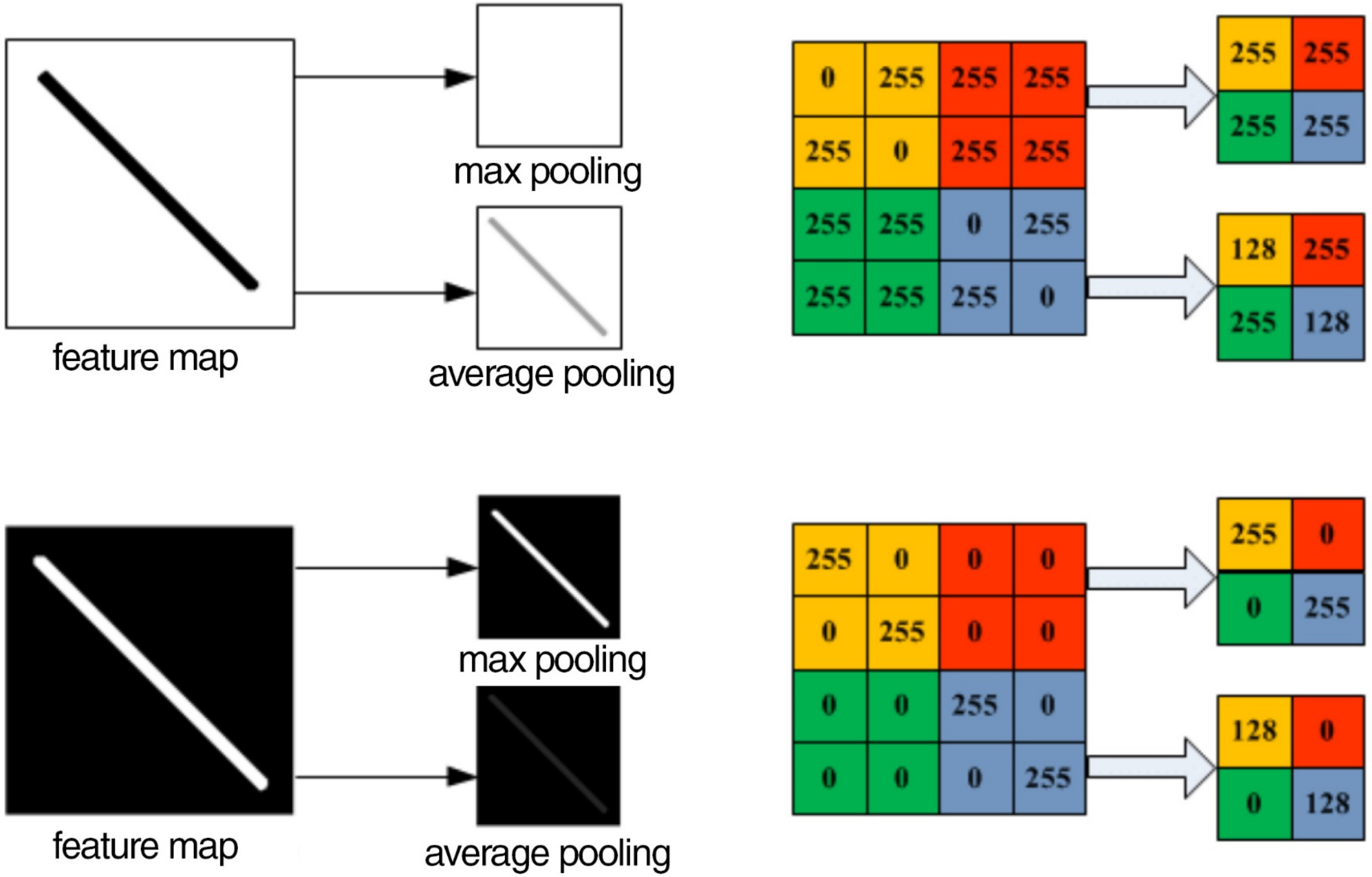


Dingjun Yu, Hanli Wang, Peiqiu Chen & Zhihua Wei

Mixed Pooling for Convolutional Neural Networks

International Conference on Rough Sets and Knowledge Technology - 2014

Pooling

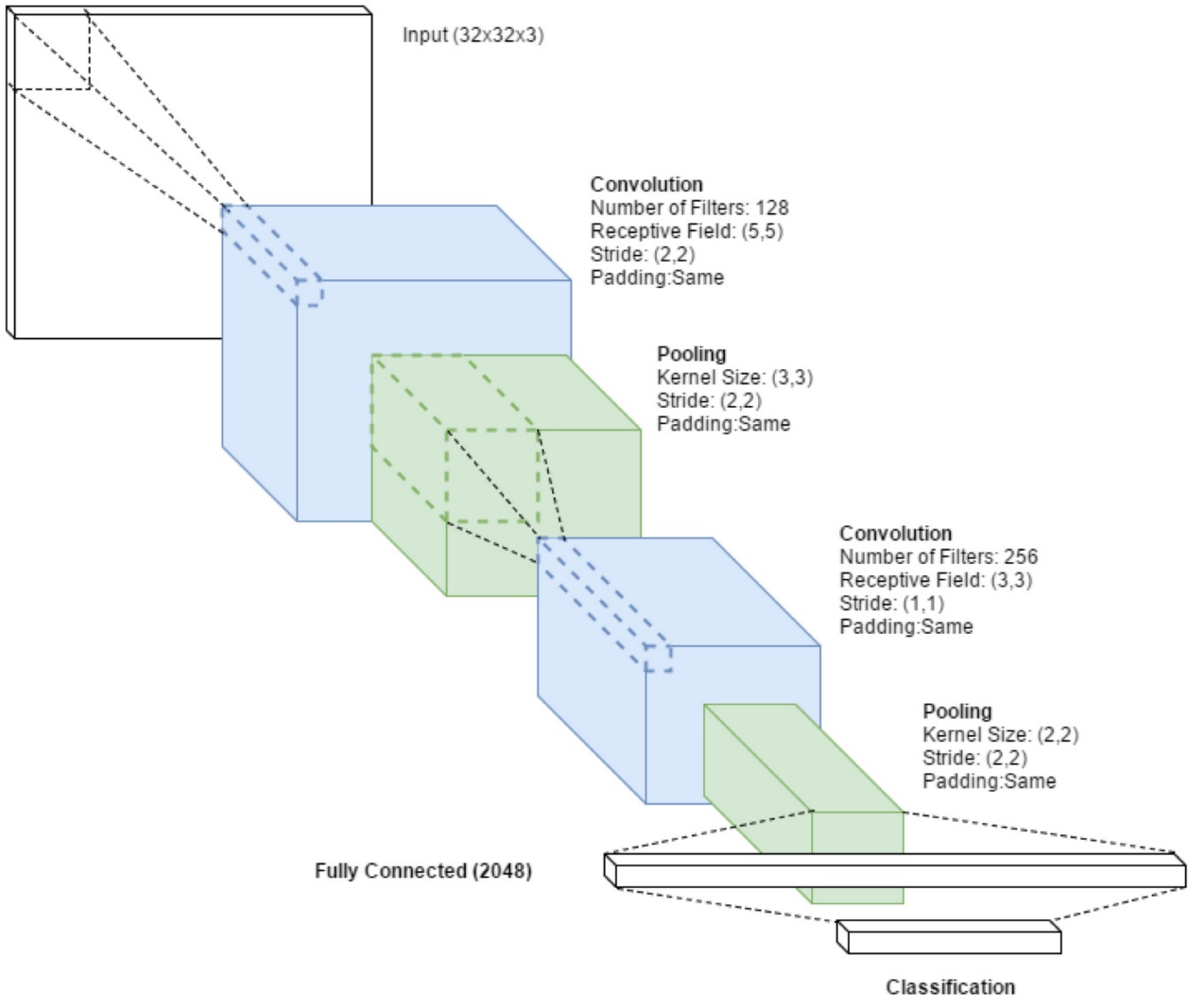


Dingjun Yu, Hanli Wang, Peiqiu Chen & Zhihua Wei

Mixed Pooling for Convolutional Neural Networks

International Conference on Rough Sets and Knowledge Technology - 2014

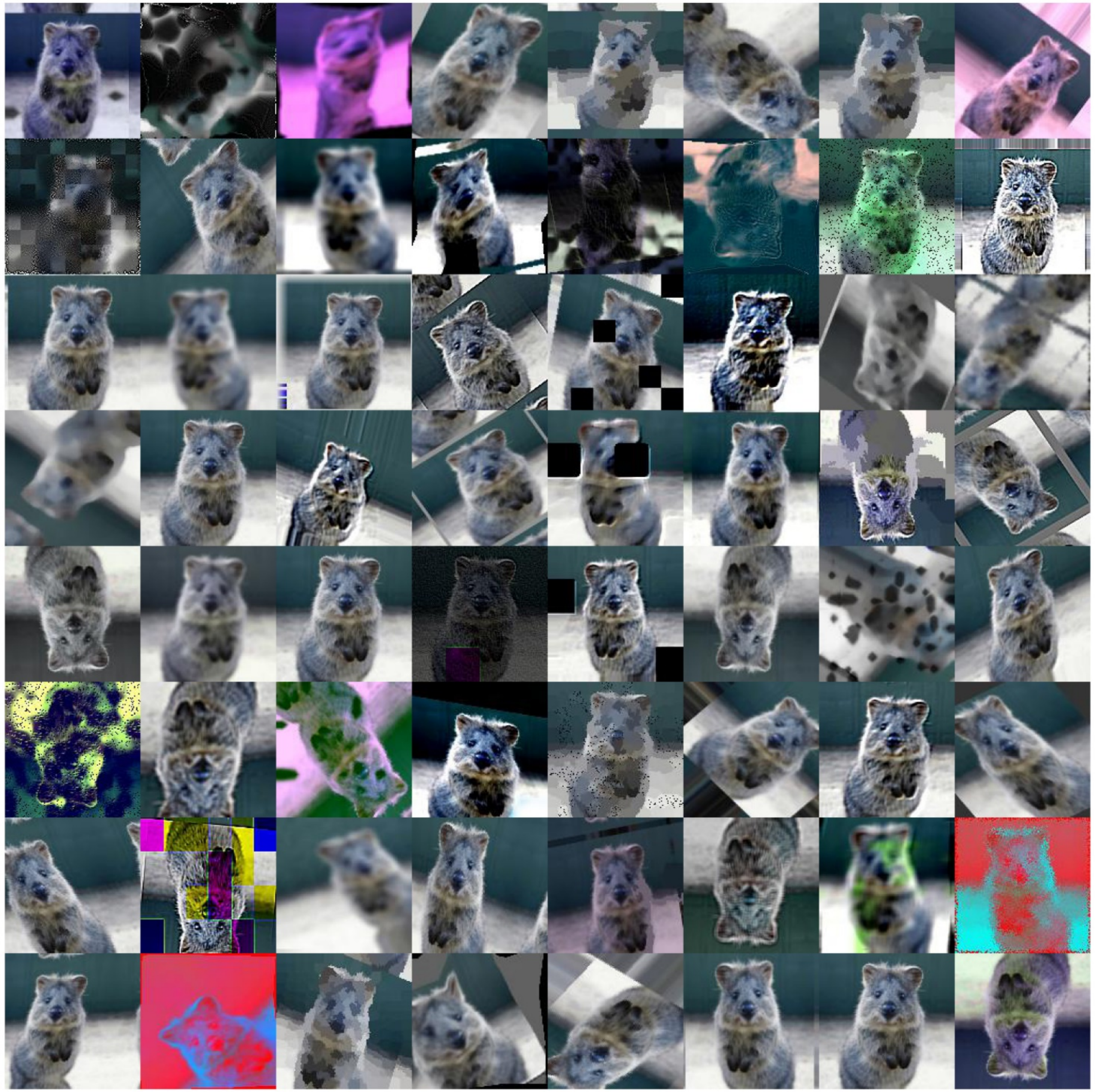
Convolutional Neural Networks



http://www.thushv.com/computer_vision/convolutional-neural-networks-mayor-of-the-visionville/

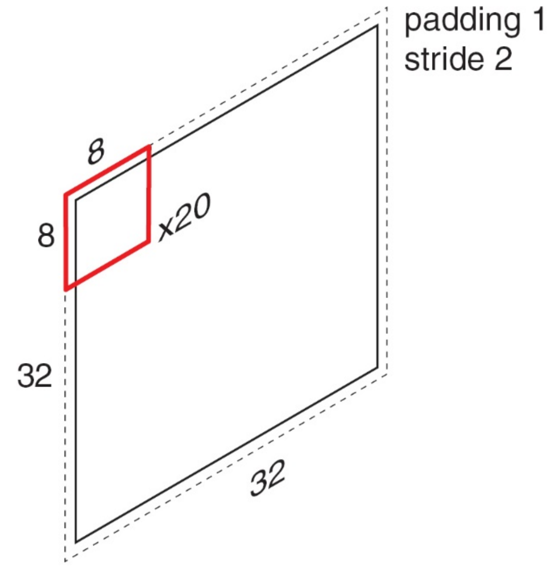
Data Augmentation

rotate
translate
shear
flip
scale/zoom
crop
apply whitening
apply noise
shift channel
shift brightness/contrast
blur
...

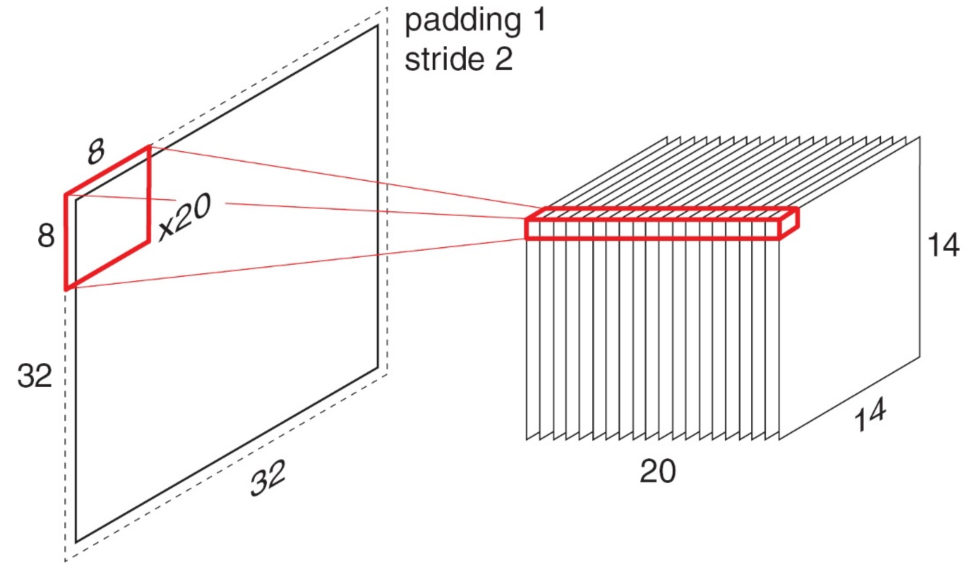


https://github.com/aleju/imgaug

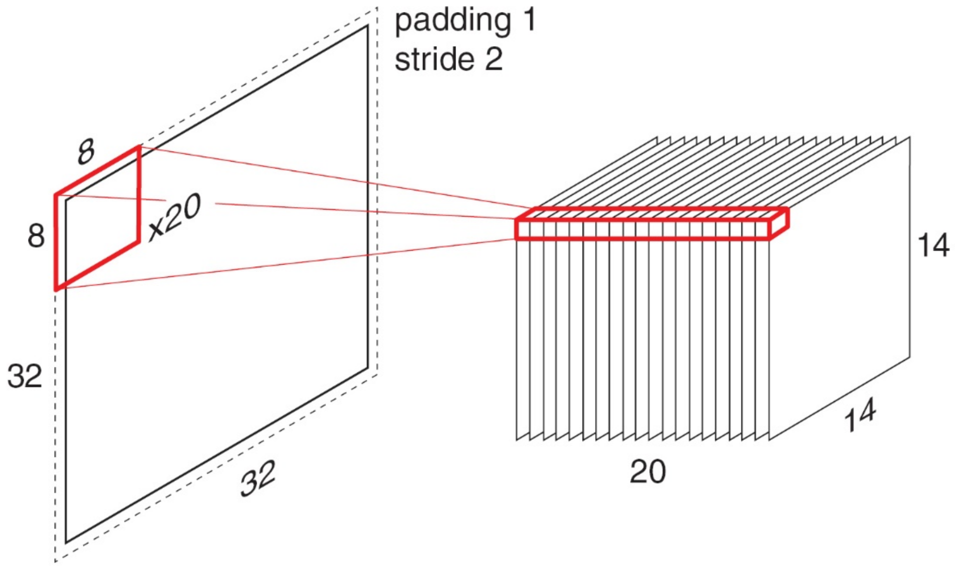
Number of Parameters



Number of Parameters

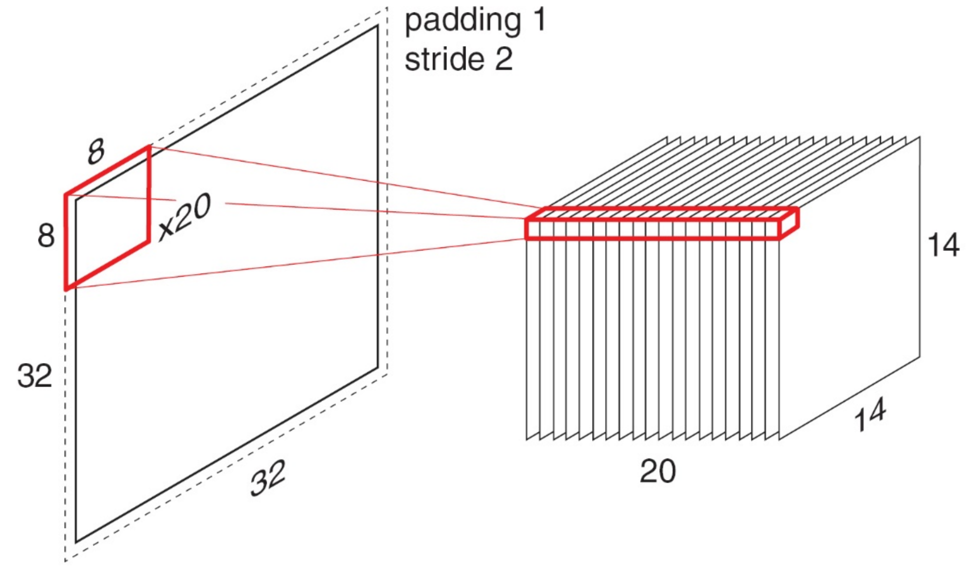


Number of Parameters



	filter		feature map				
without weight sharing	(8*8*1 + 1)	*	(14*14)	*	20	=	**254,800**

Number of Parameters



	filter		feature map				
without weight sharing	(8*8*1 + 1)	*	(14*14)	*	20	=	**254,800**
with weight sharing	(8*8*1 + 1)		*		20	=	**1,300**

Memory Management

INPUT: [224x224x3]	memory: 224*224*3=150K	weights: 0
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	weights: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	weights: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28*256=200K	weights: 0
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	weights: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	weights: 0
FC: [1x1x4096]	memory: 4096	weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]	memory: 4096	weights: 4096*4096 = 16,777,216
FC: [1x1x1000]	memory: 1000	weights: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB
TOTAL params: 138M parameters

Karen Simonyan & Andrew Zisserman

Very Deep Convolutional Networks for Large-Scale Image Recognition
International Conference on Rough Sets and Knowledge Technology - 2014

Memory Management

INPUT: [224x224x3]	memory: 224*224*3=150K	weights: 0
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	memory: 224*224*64=3.2M	weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]	memory: 112*112*64=800K	weights: 0
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]	memory: 112*112*128=1.6M	weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]	memory: 56*56*128=400K	weights: 0
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]	memory: 56*56*256=800K	weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]	memory: 28*28*256=200K	weights: 0
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	memory: 28*28*512=400K	weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]	memory: 14*14*512=100K	weights: 0
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	memory: 14*14*512=100K	weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	memory: 7*7*512=25K	weights: 0
FC: [1x1x4096]	memory: 4096	weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]	memory: 4096	weights: 4096*4096 = 16,777,216
FC: [1x1x1000]	memory: 1000	weights: 4096*1000 = 4,096,000

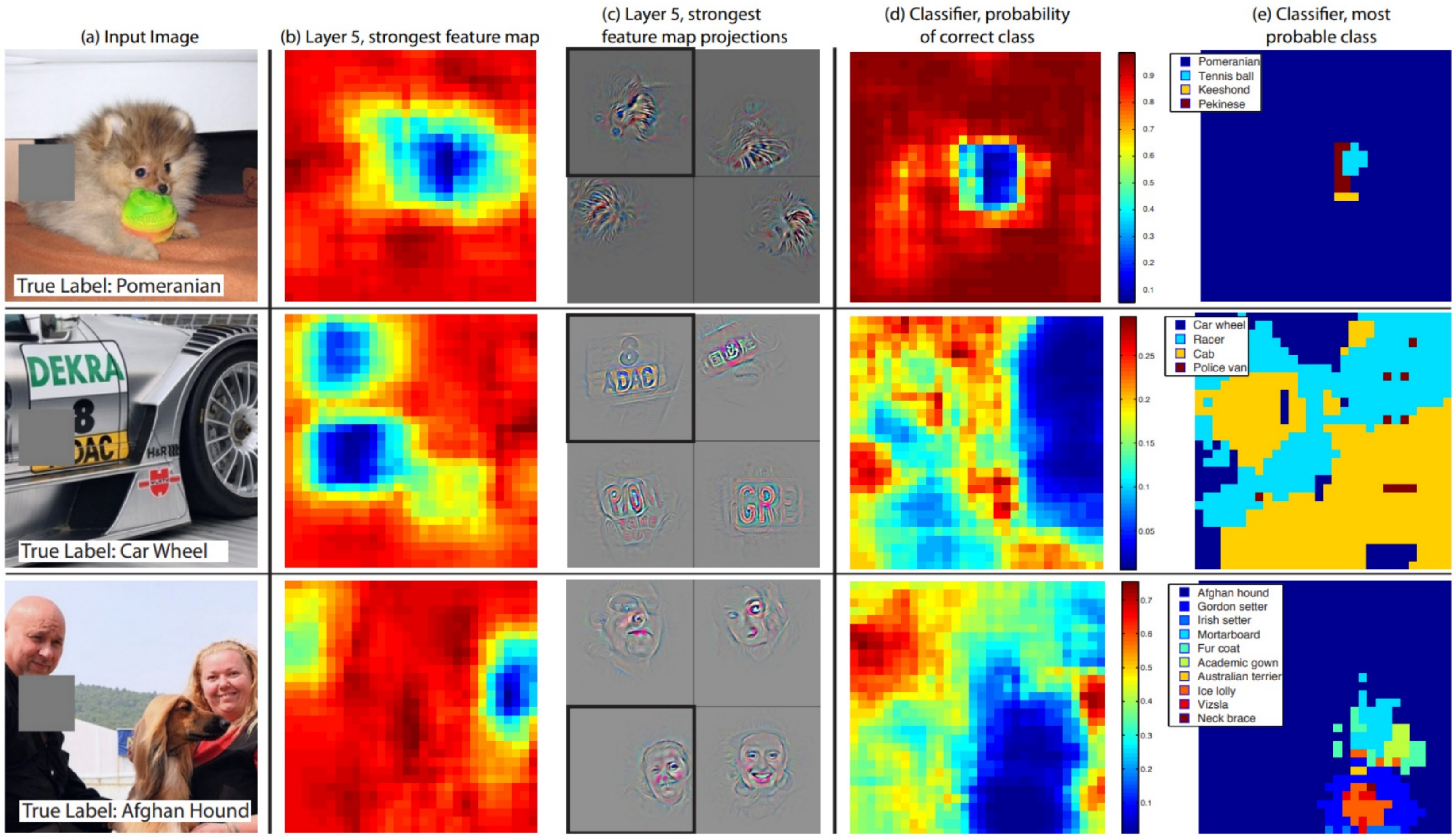
TOTAL memory: 24M * 4 bytes ~= 93MB
TOTAL params: 138M parameters

Nvidia GeForce GTX 980 4GB

Karen Simonyan & Andrew Zisserman

Very Deep Convolutional Networks for Large-Scale Image Recognition
International Conference on Rough Sets and Knowledge Technology - 2014

Visualizing Attention

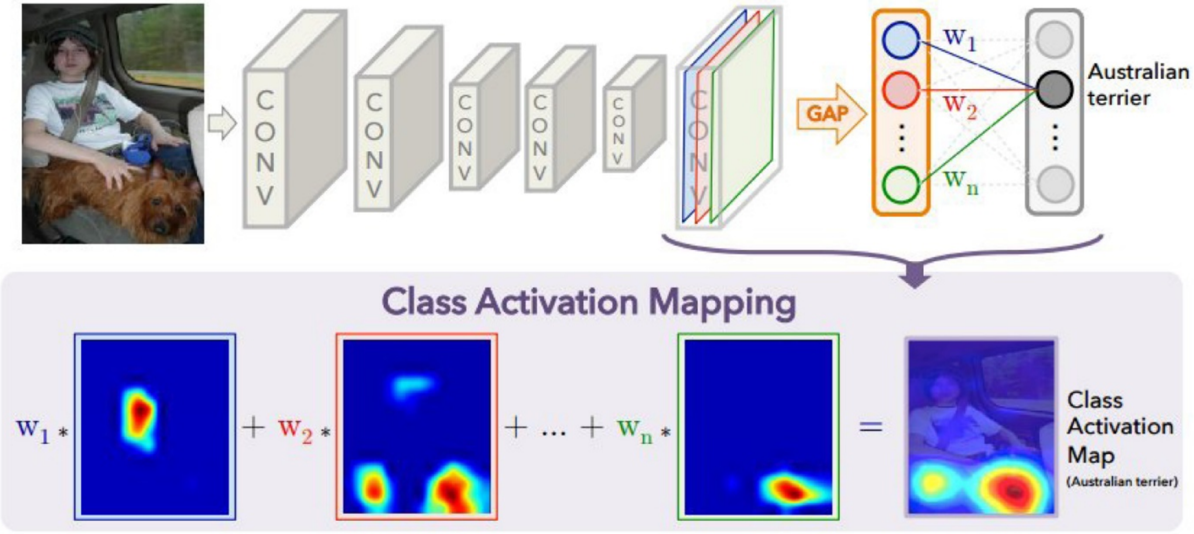


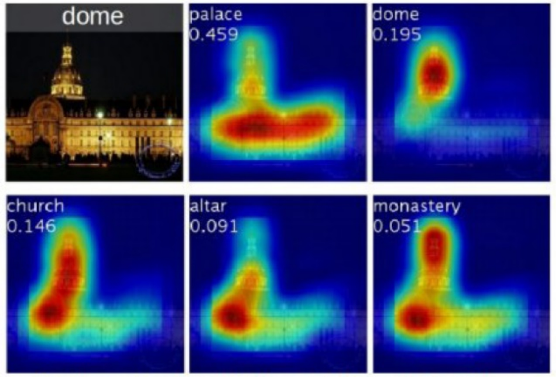
Matthew Zeiler & Rob Fergus

Visualizing and Understanding Convolutional Networks

European Conference on Computer Vision - 2014

Visualizing Attention





Class activation maps of top 5 predictions

Class activation maps for one object class

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva & Antonio Torralba

Learning Deep Features for Discriminative Localization

Conference on Computer Vision and Pattern Recognition - 2016

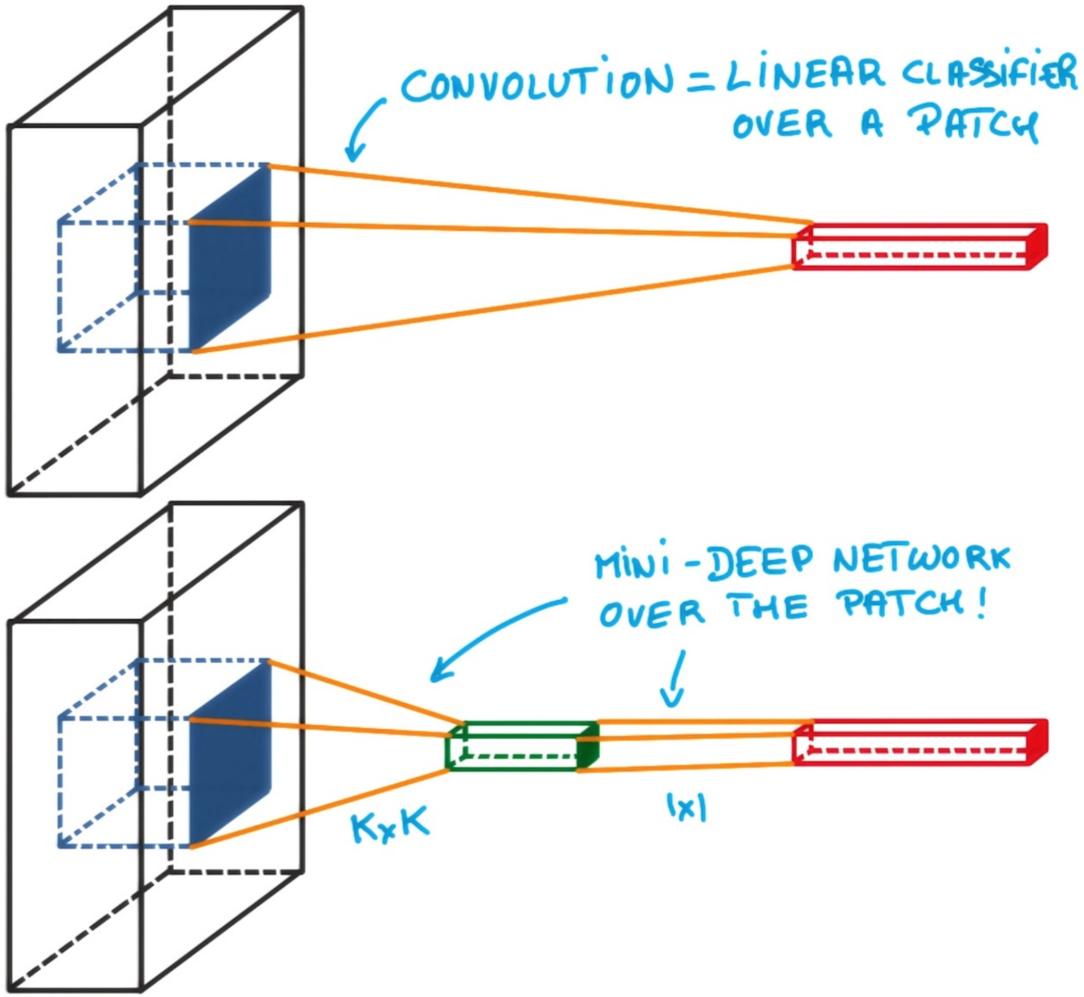
Why do we need convolutions?

CNN specifics

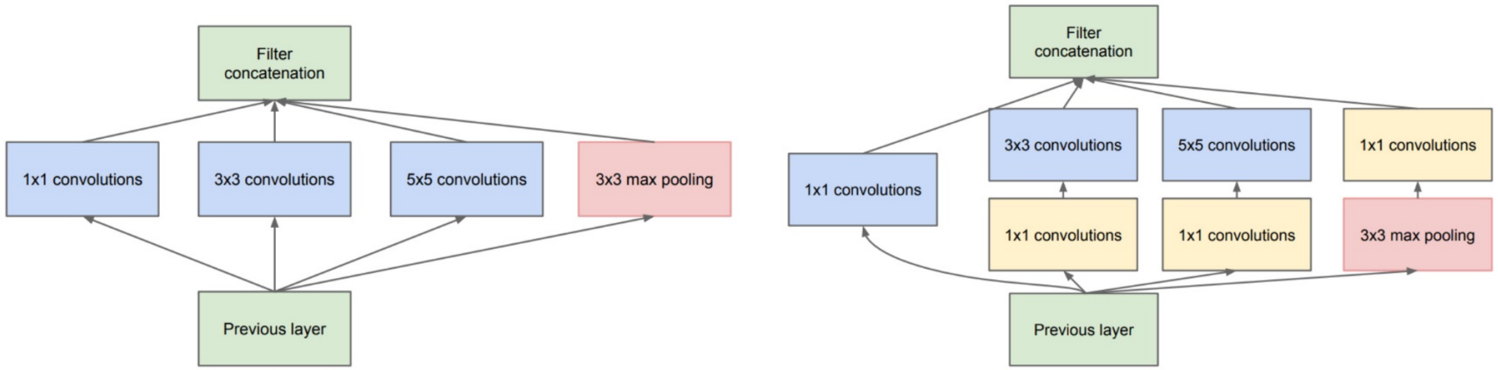
CNN flavors

Resources

1x1 Convolutions



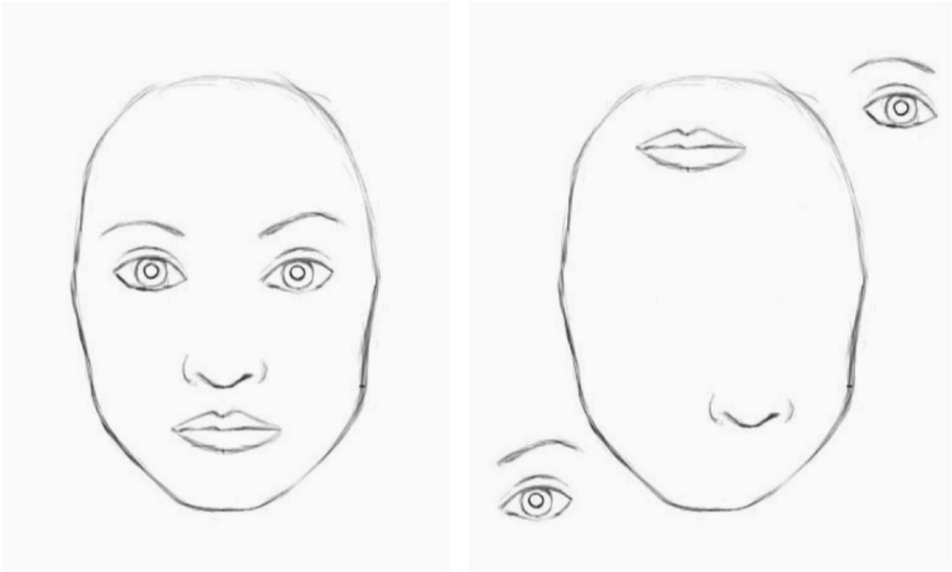
Inception Module



(a) Inception module, naïve version (b) Inception module with dimension reductions

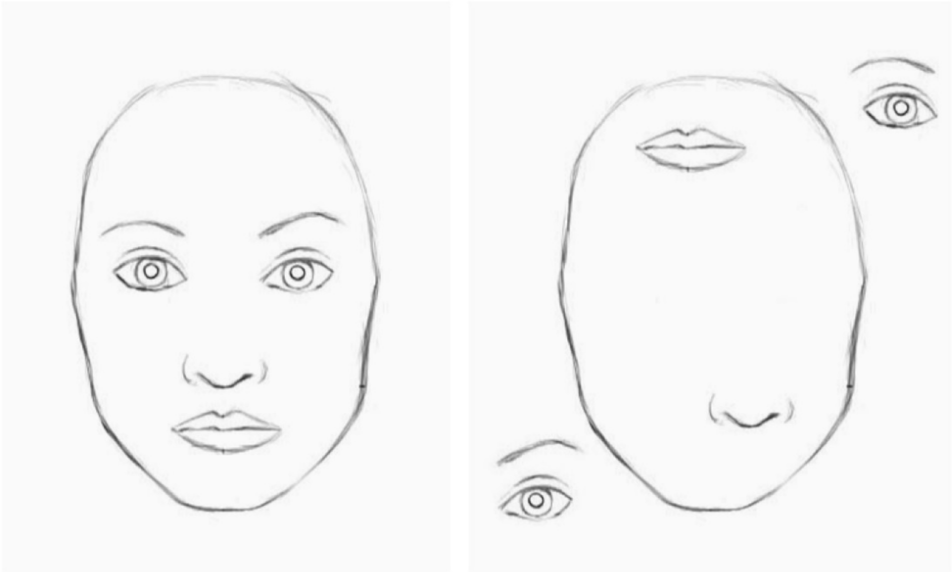
Figure 2: Inception module

Capsule Networks



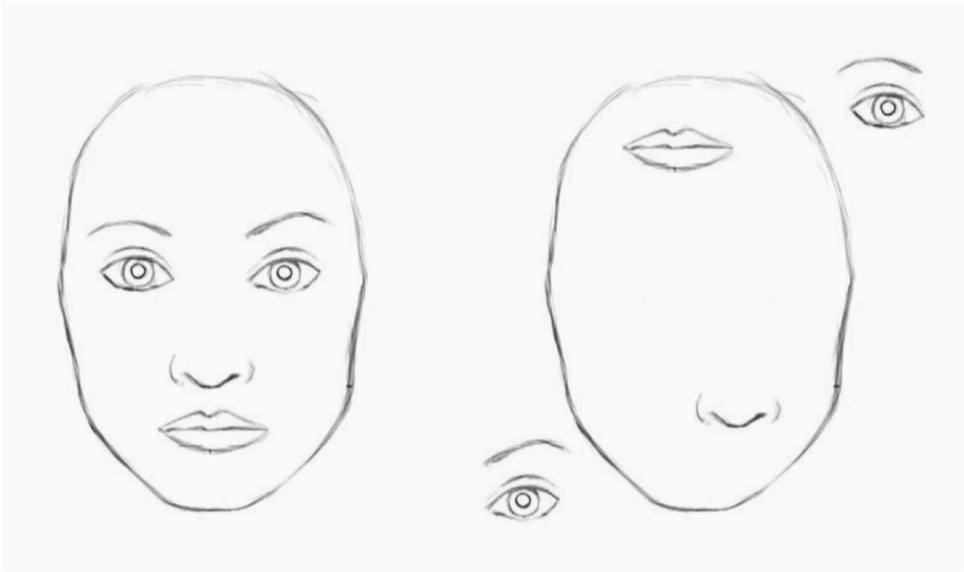
https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Capsule Networks



https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Capsule Networks

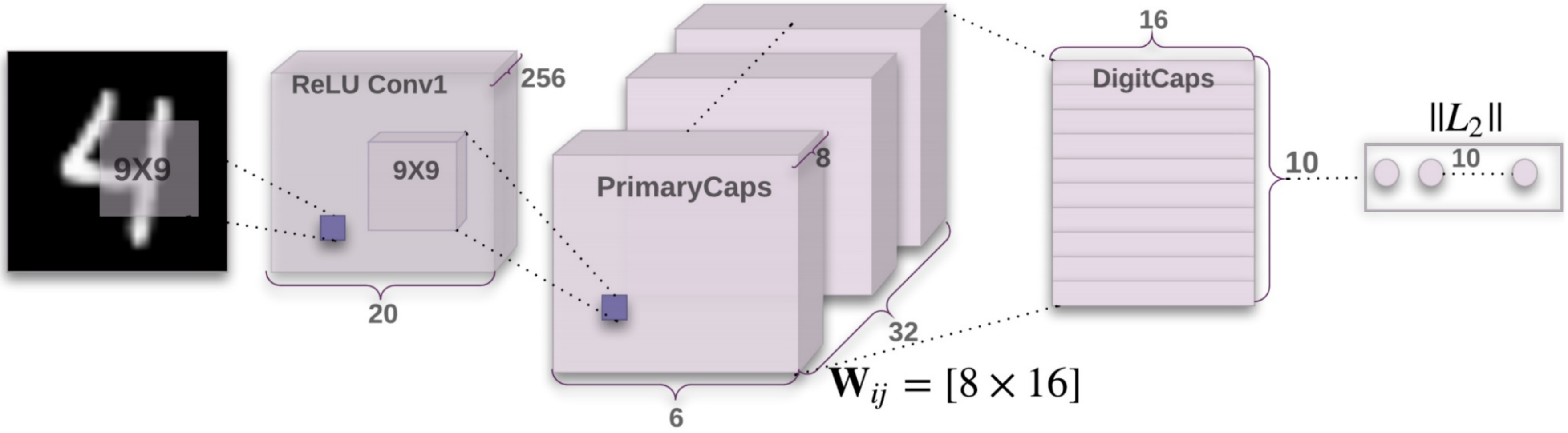


"The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster." **-- Geoffrey Hinton**

https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b

Capsule Networks

Figure 1: A simple CapsNet with 3 layers. This model gives comparable results to deep convolutional networks (such as Chang and Chen [2015]). The length of the activity vector of each capsule in DigitCaps layer indicates presence of an instance of each class and is used to calculate the classification loss. $\mathbf{W}_{ij}$ is a weight matrix between each $\mathbf{u}_i, i \in (1, 32 \times 6 \times 6)$ in PrimaryCapsules and $\mathbf{v}_j, j \in (1, 10)$.

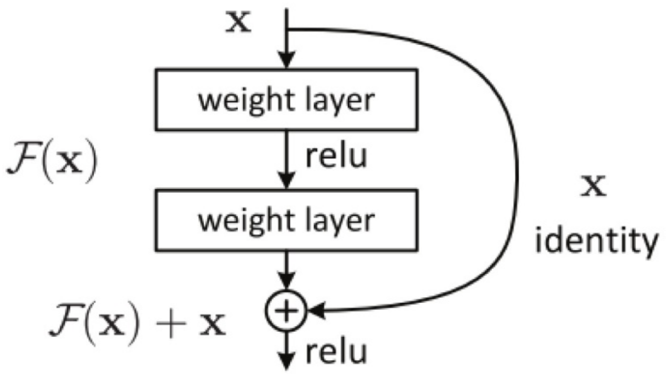


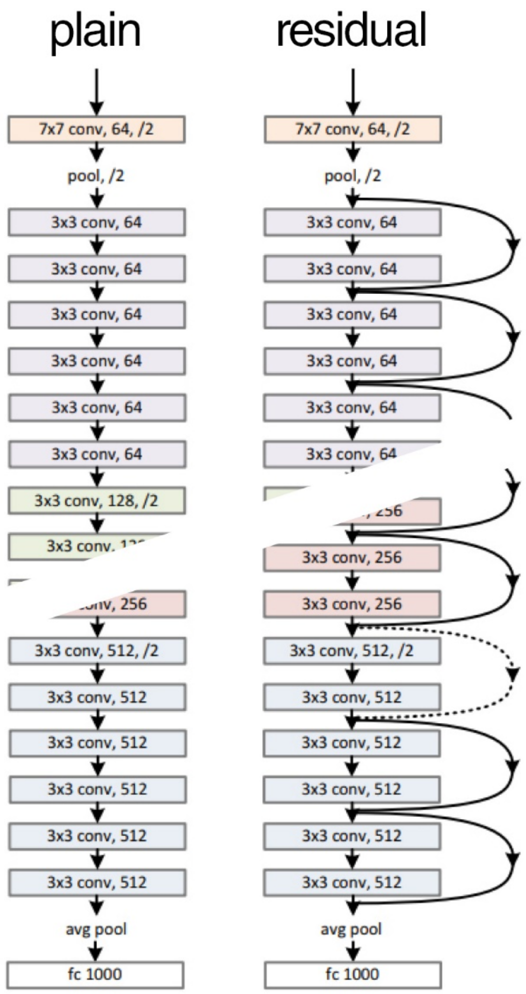
Sara Sabour, Nicholas Frosst & Geoffrey Hinton

Dynamic Routing Between Capsules

Conference on Neural Information Processing Systems - 2017

ResNets

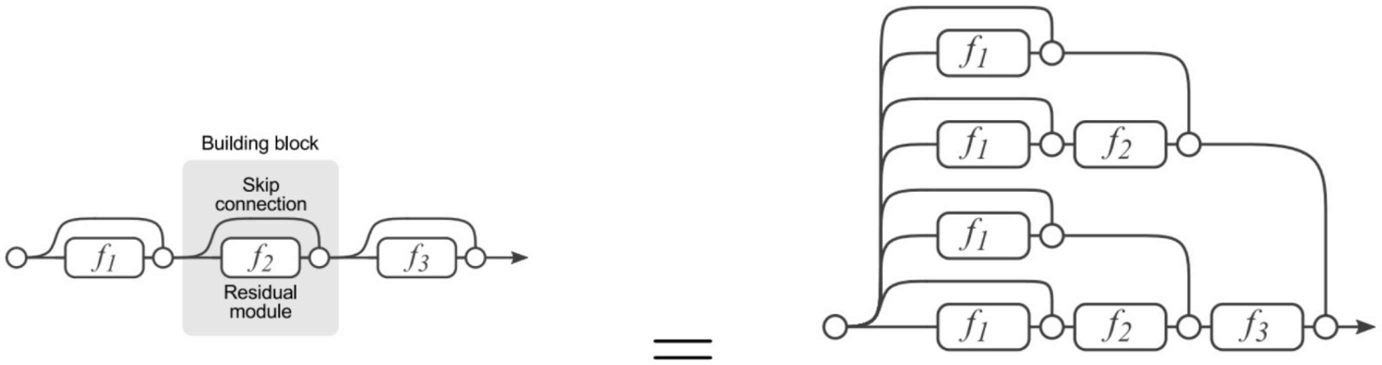




Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun

Deep Residual Learning for Image Recognition

Conference on Computer Vision and Pattern Recognition - 2016



(a) Conventional 3-block residual network

(b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

Andreas Veit, Michael Wilber & Serge Belongie

Residual Networks Behave Like Ensembles of Relatively Shallow Networks

Conference on Computer Vision and Pattern Recognition - 2016

Fully Convolutional Networks

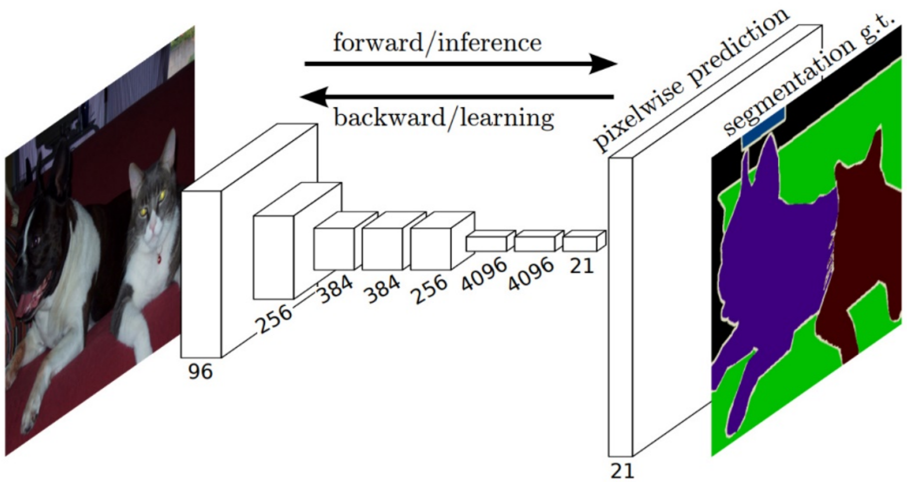


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

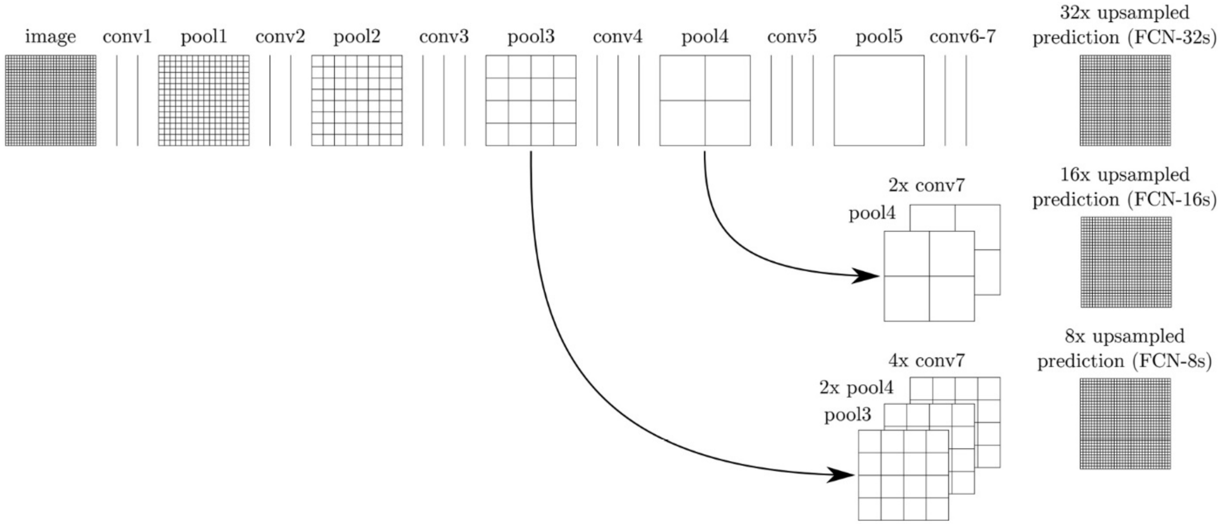


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

Jonathan Long, Evan Shelhamer & Trevor Darrell

Fully Convolutional Networks for Semantic Segmentation

CVPR - 2015

Fully Convolutional Networks

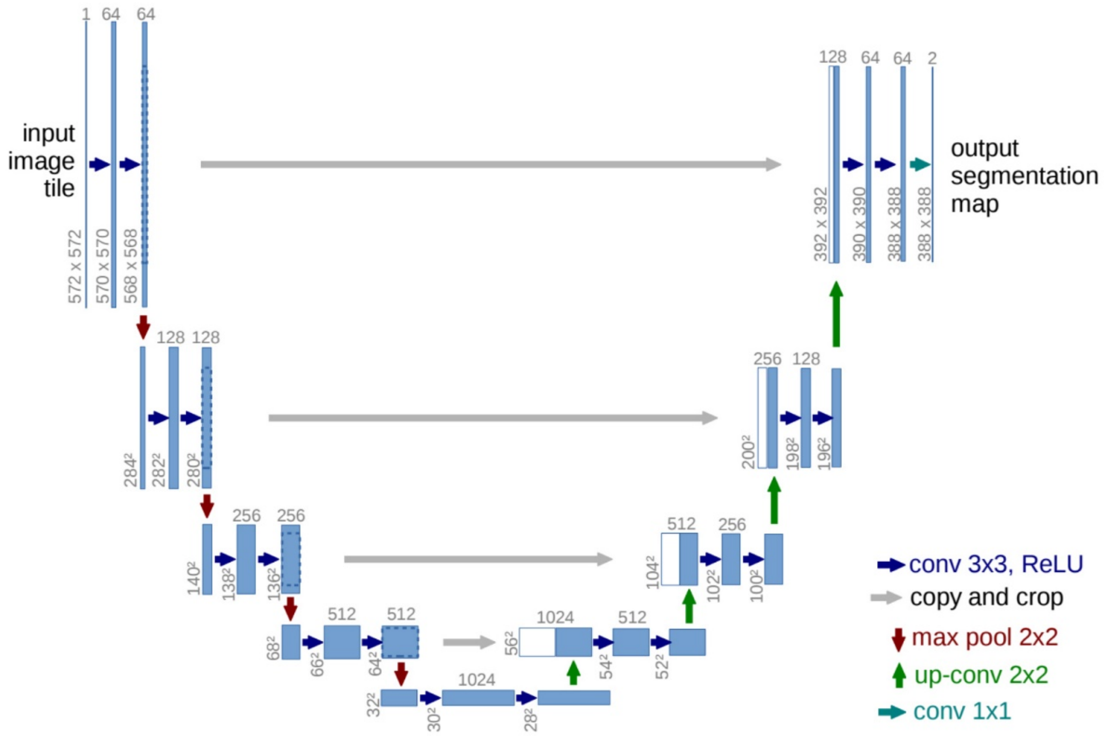


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Olaf Ronneberger, Philipp Fischer & Thomas Brox

U-Net: Convolutional Networks for Biomedical Image Segmentation

MICCAI - 2015

Fully Convolutional Networks

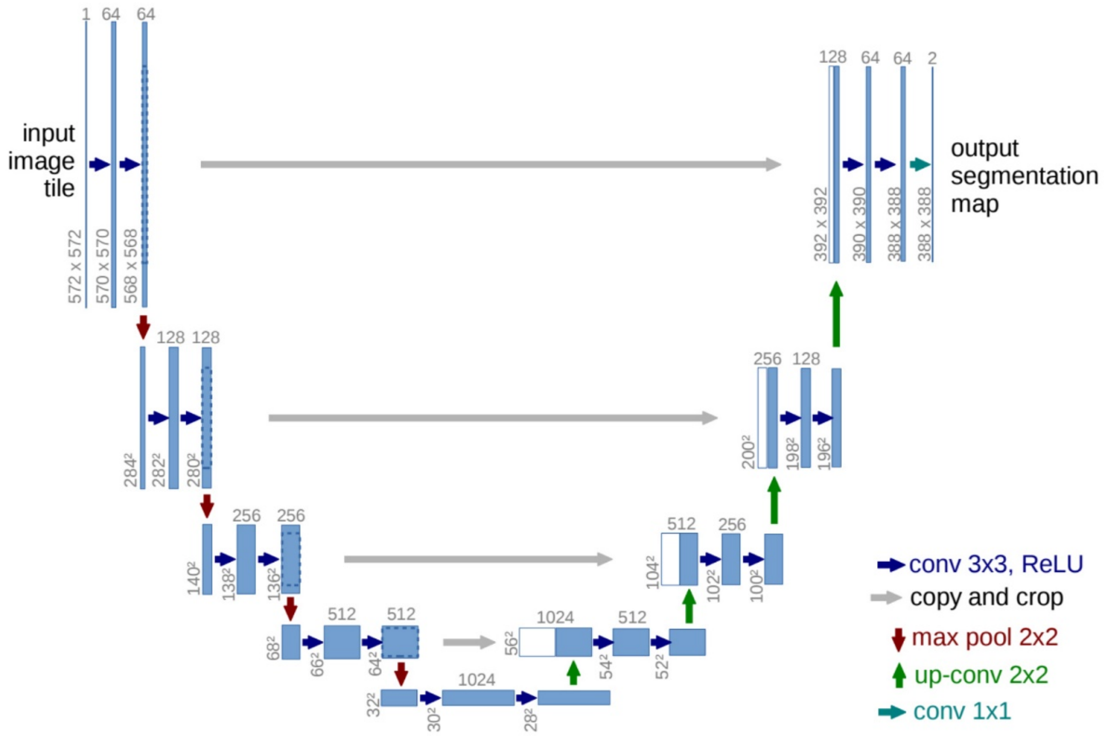


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

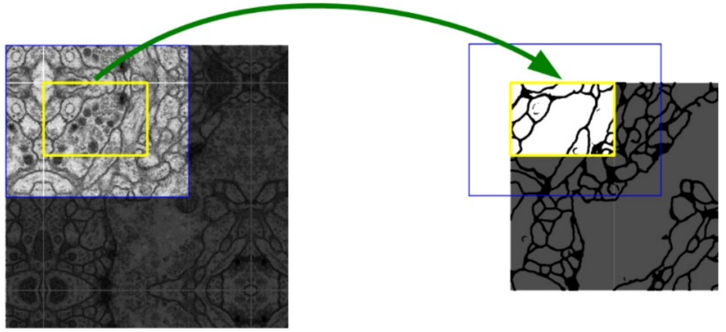


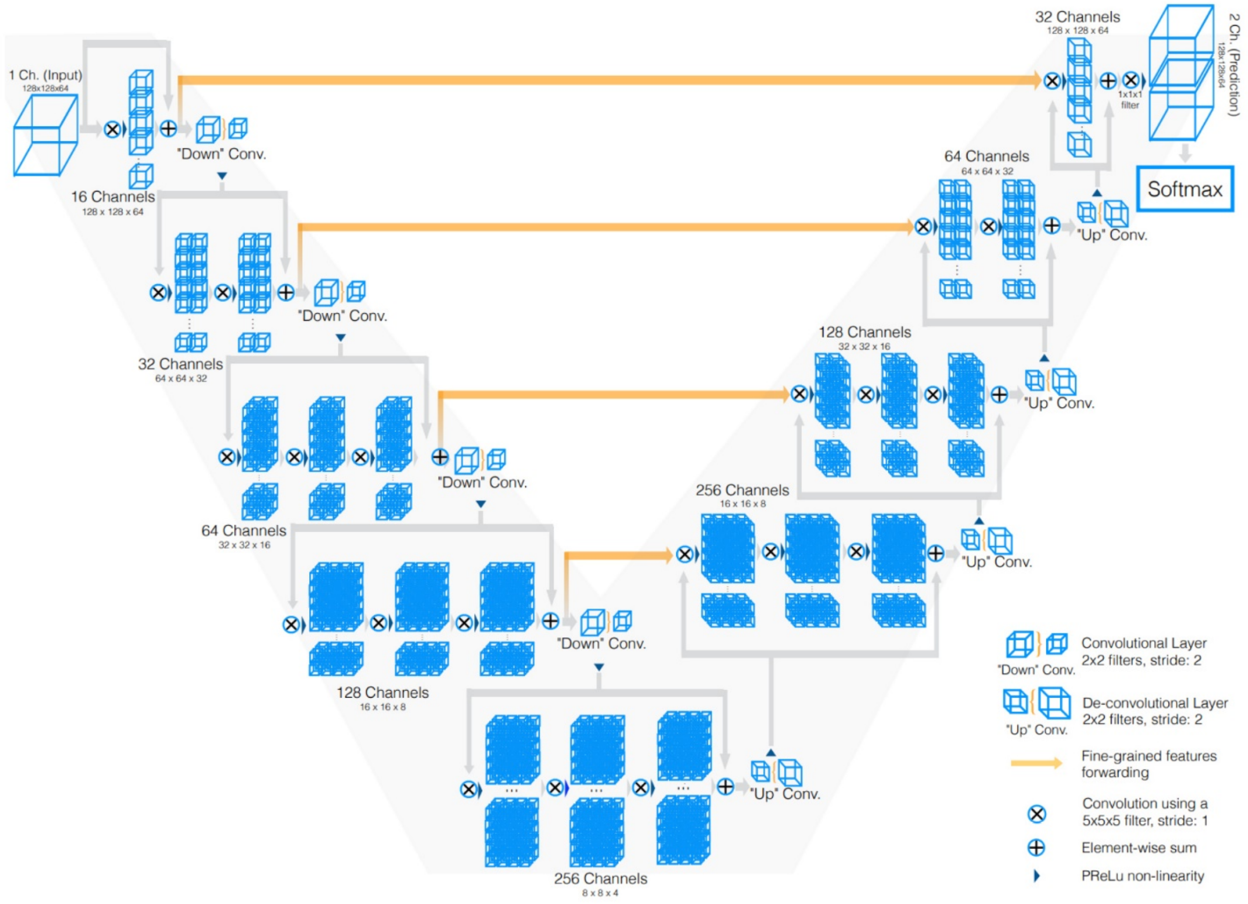
Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

Olaf Ronneberger, Philipp Fischer & Thomas Brox

U-Net: Convolutional Networks for Biomedical Image Segmentation

MICCAI - 2015

Fully Convolutional Networks

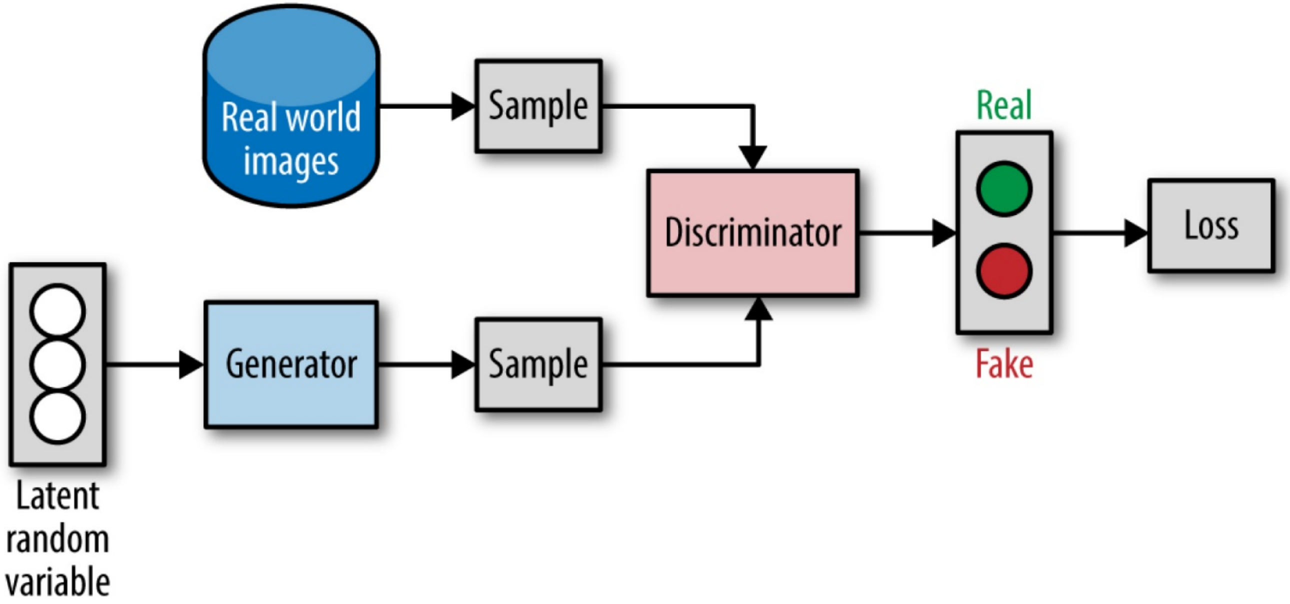


Fausto Milletari, Nassir Navab & Seyed-Ahmad Ahmadi

V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation

https://arxiv.org/abs/1606.04797

Generative Adversarial Networks (GAN)



https://www.safaribooksonline.com/library/view/tensorflow-for-deep/9781491980446/ch01.html

Generative Adversarial Networks (GAN)

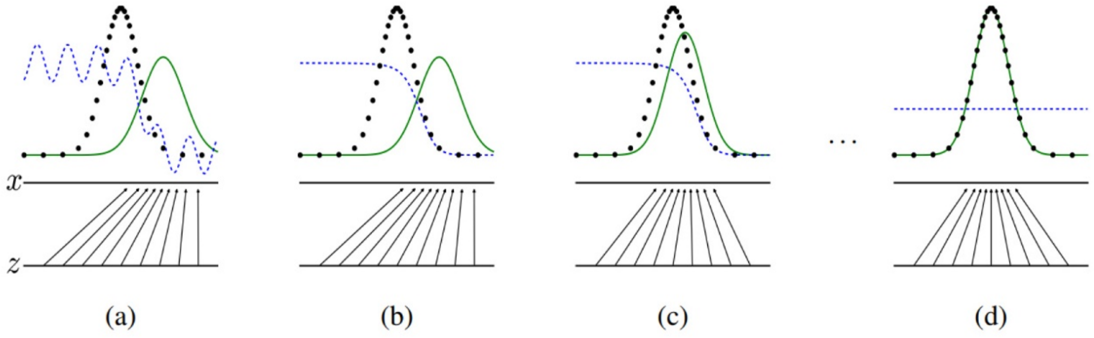


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville & Yoshua Bengio

Generative Adversarial Nets

Advances in Neural Information Processing Systems - 2014

Deep Convolutional Generative Adversarial Networks (DCGAN)

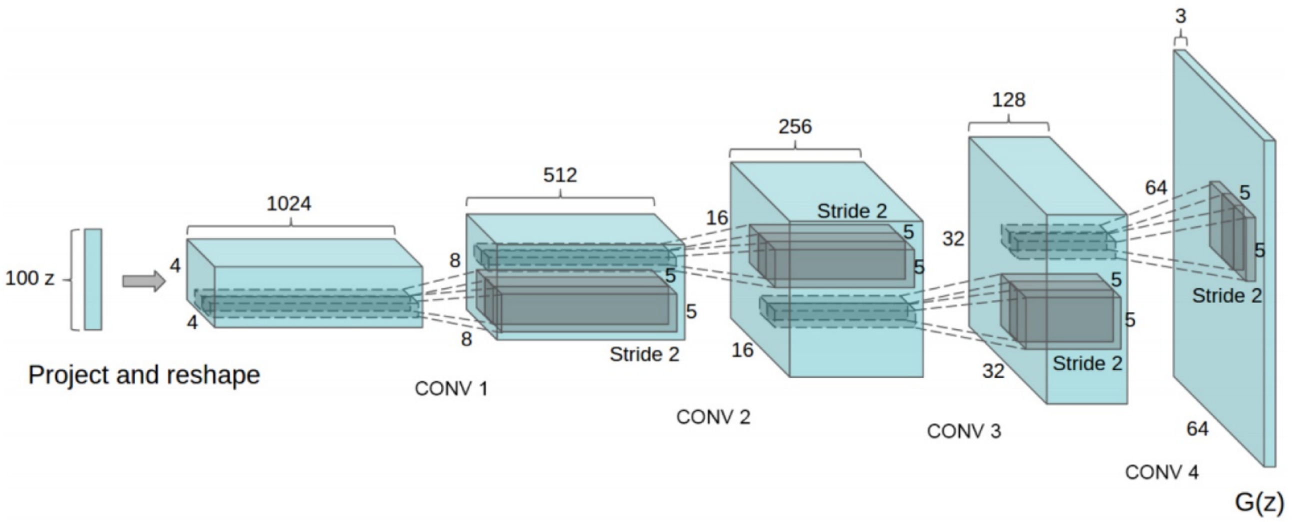


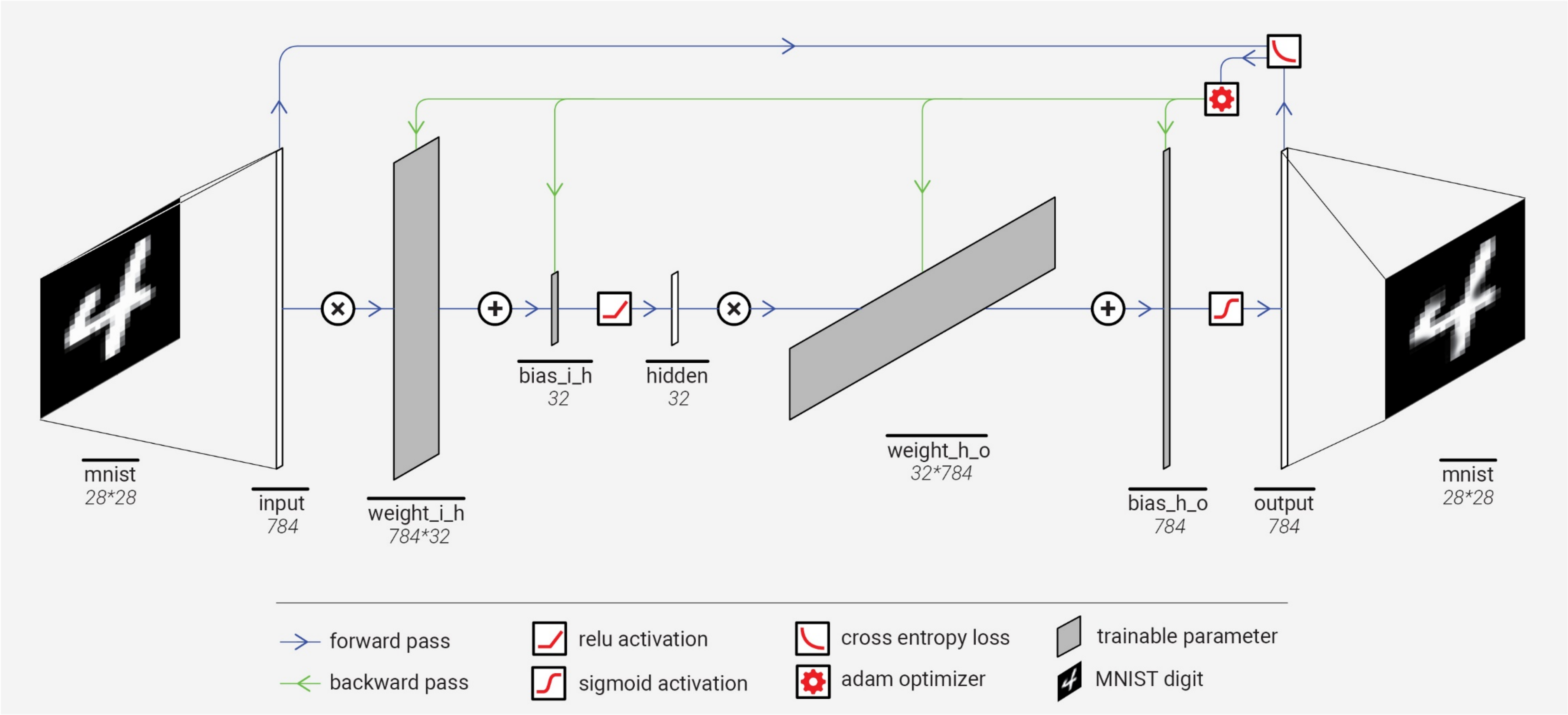
Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

Alec Radford, Luke Metz & Soumith Chintala

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

International Conference on Learning Representations - 2016

Variational Autoencoders



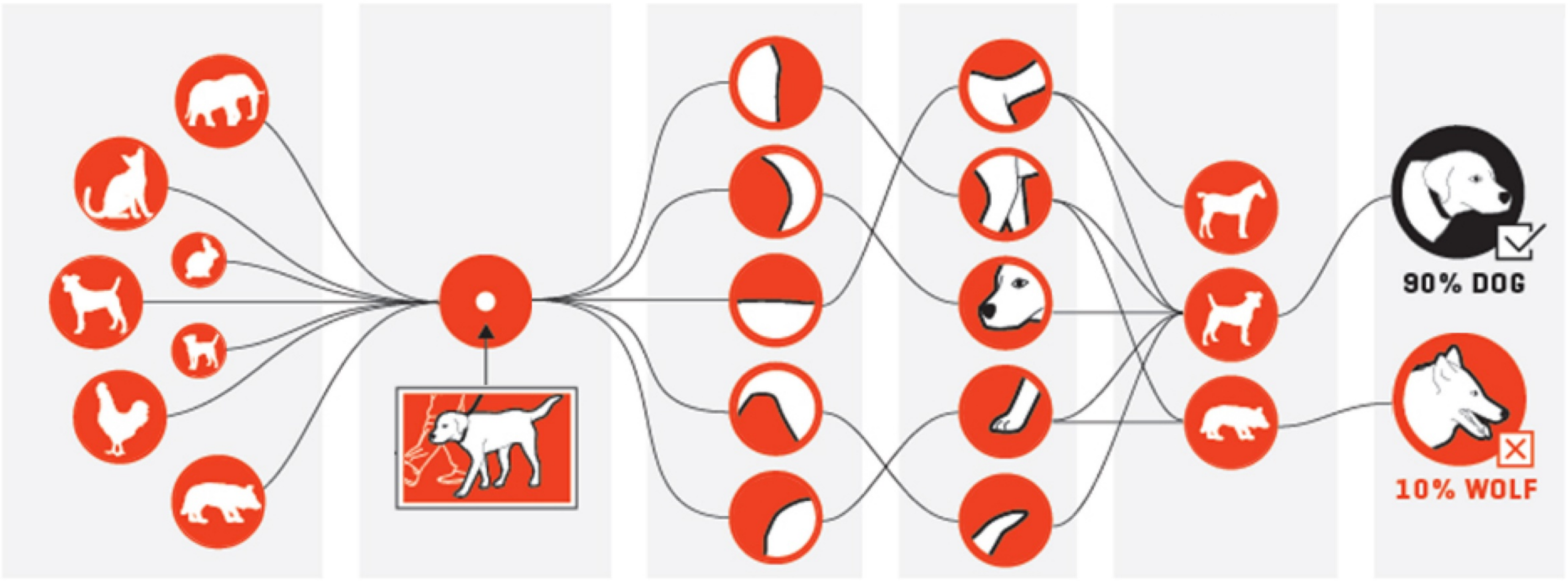
Why do we need convolutions?

CNN specifics

CNN flavors

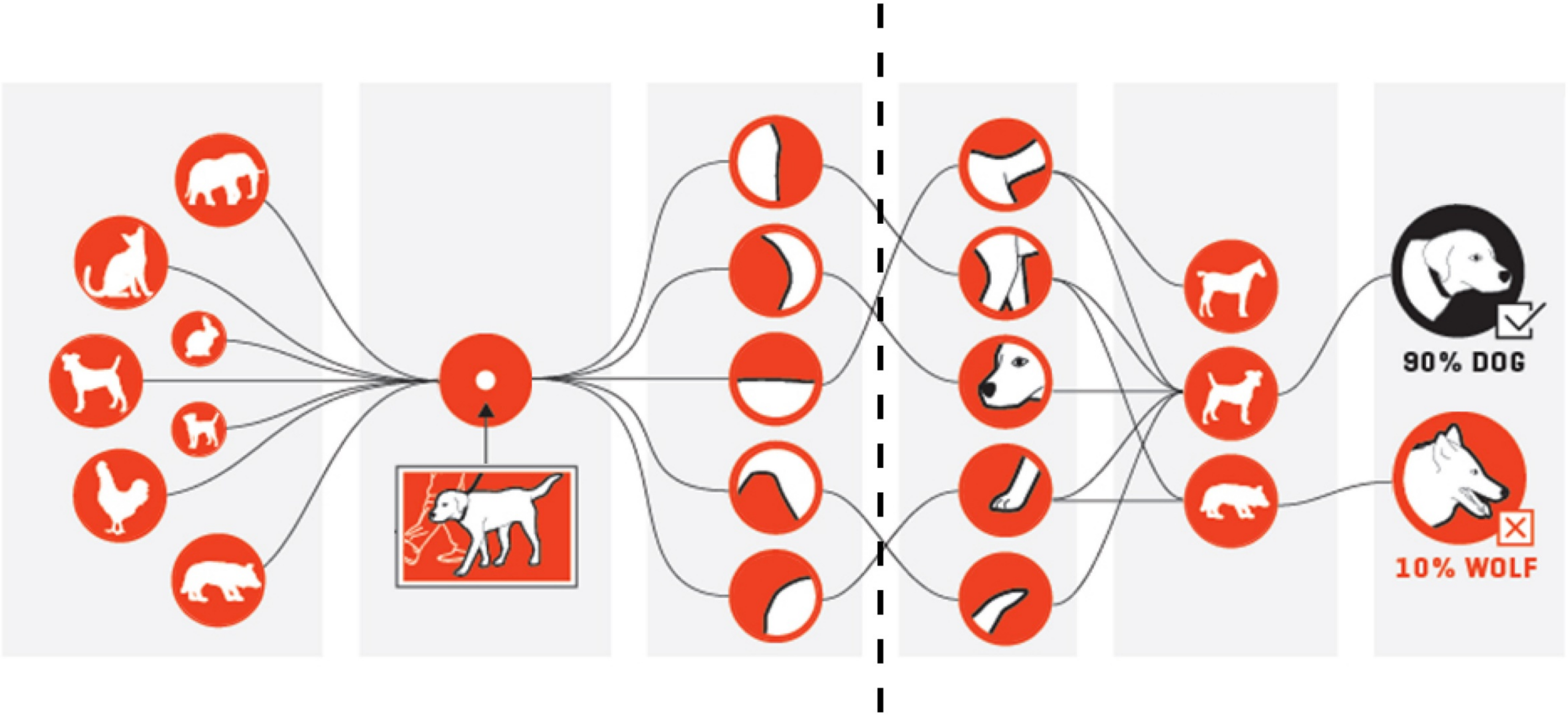
Resources

Transfer Learning



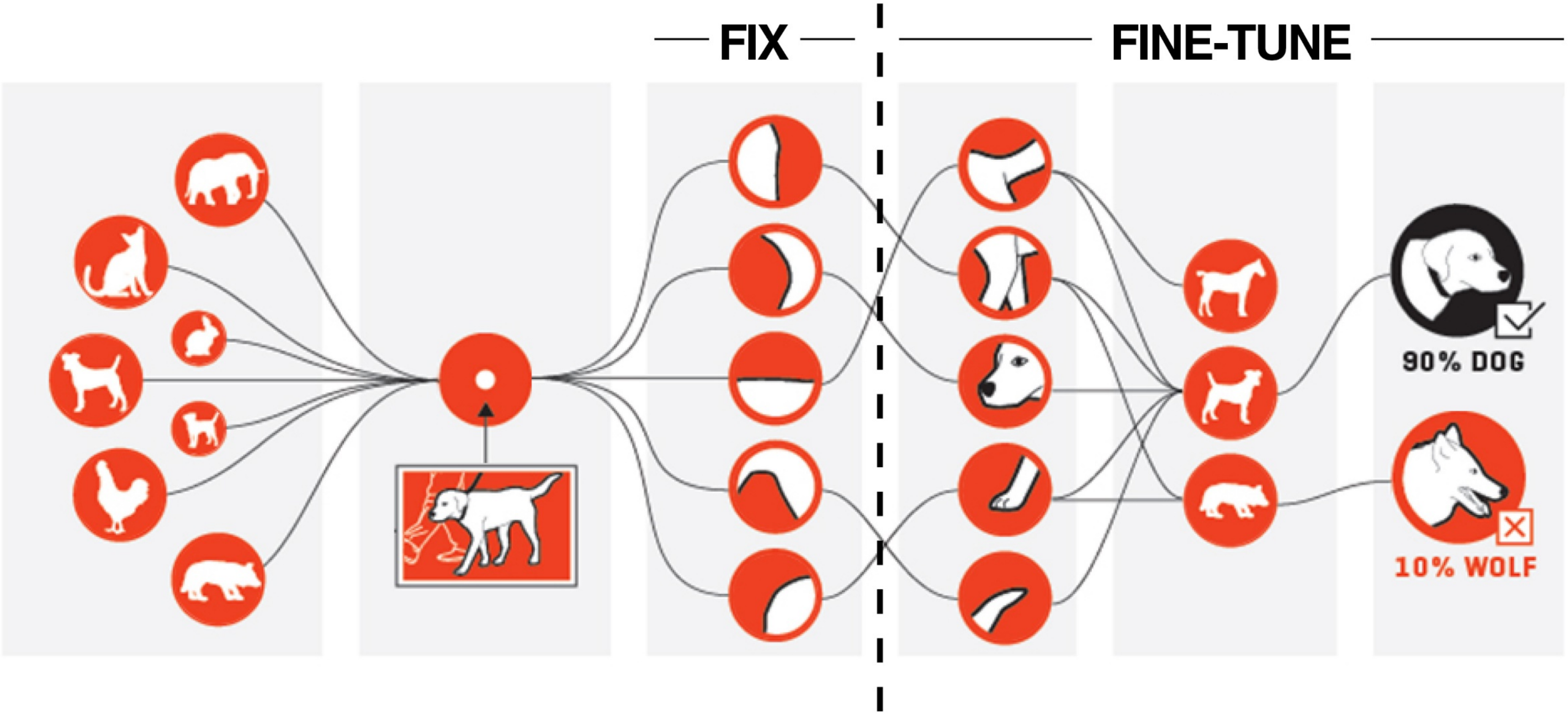
http://fortune.com/ai-artificial-intelligence-deep-machine-learning/

Transfer Learning



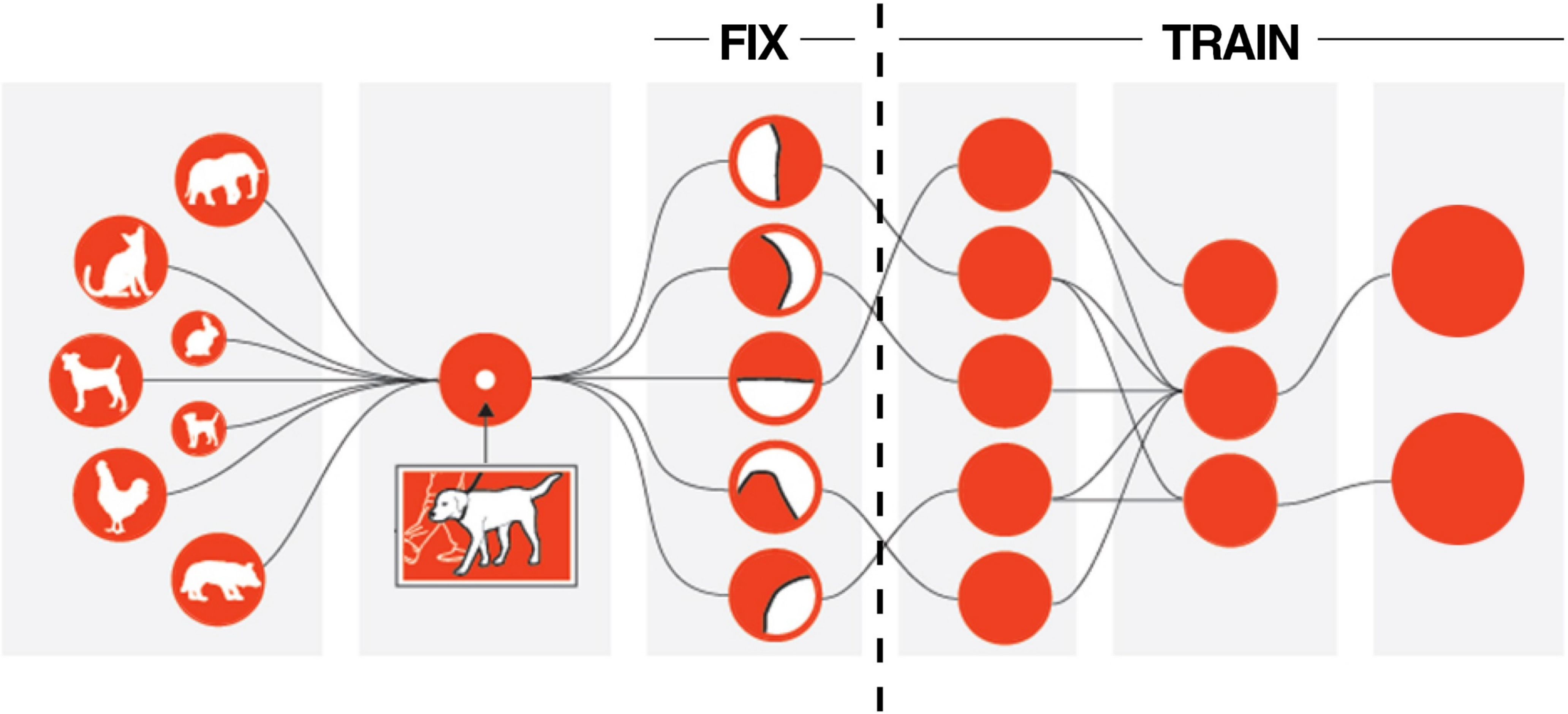
http://fortune.com/ai-artificial-intelligence-deep-machine-learning/

Transfer Learning



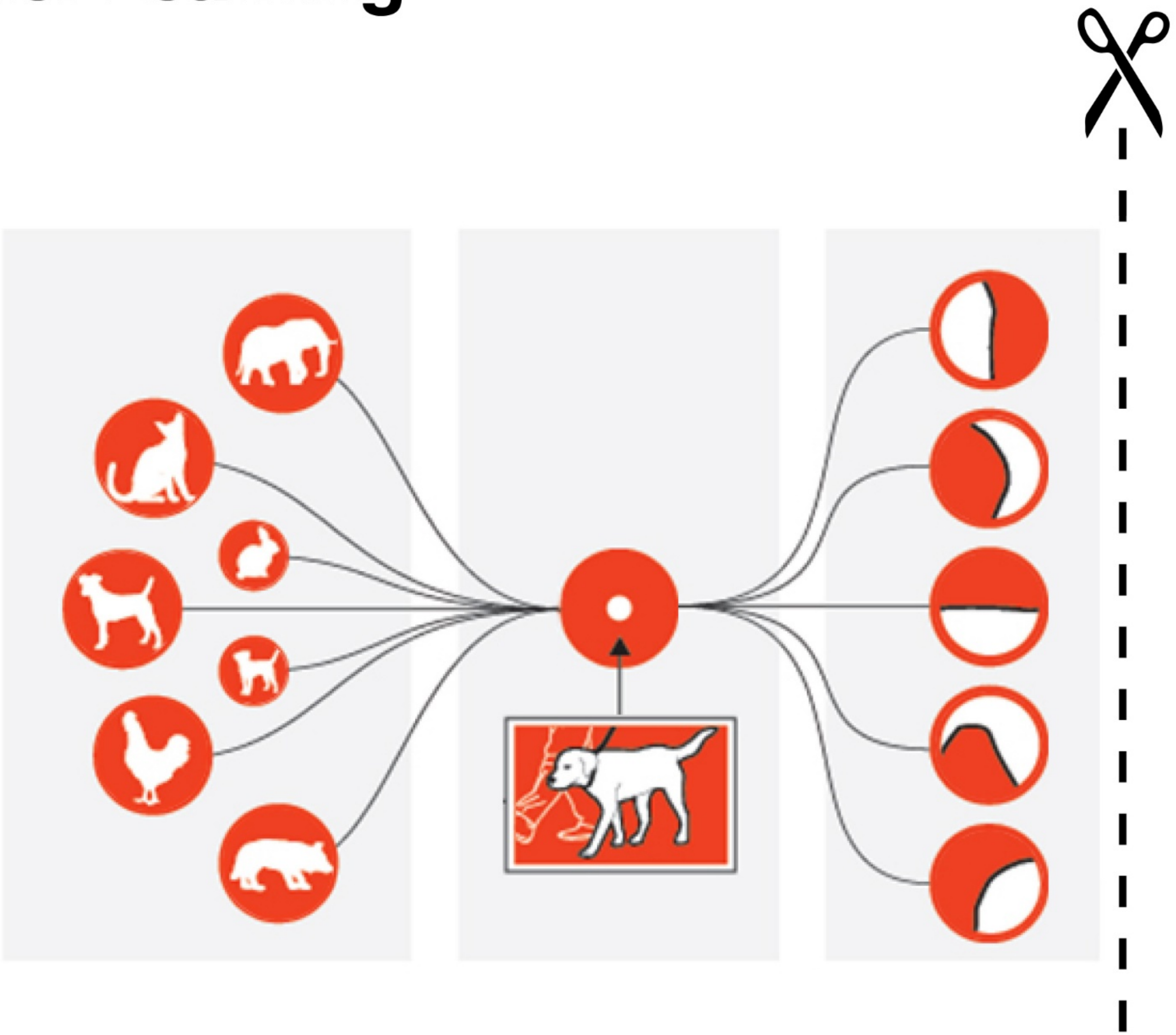
http://fortune.com/ai-artificial-intelligence-deep-machine-learning/

Transfer Learning



http://fortune.com/ai-artificial-intelligence-deep-machine-learning/

Transfer Learning



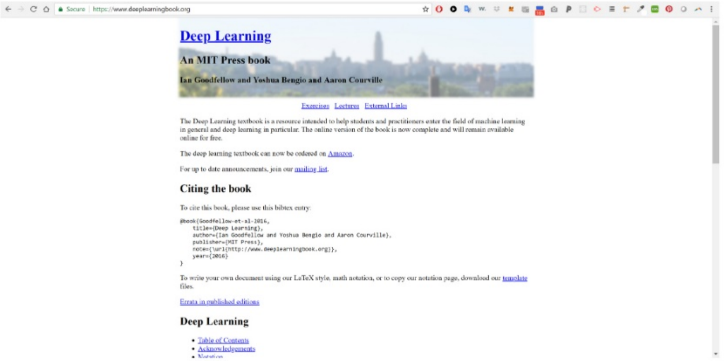
http://fortune.com/ai-artificial-intelligence-deep-machine-learning/

Transfer Learning

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

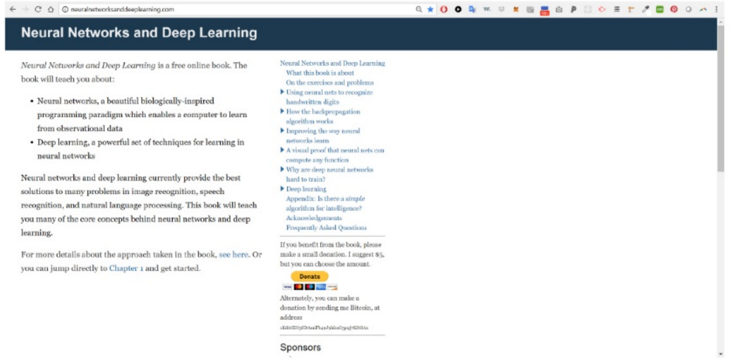
https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Online Resources



Deep Learning Book

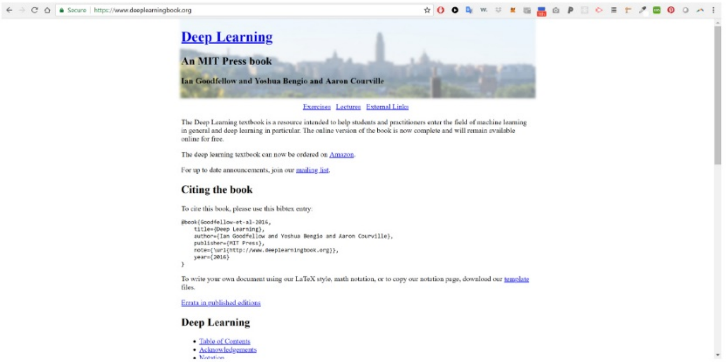
Ian Goodfellow, Yoshua Bengio, Aaron Courville



Neural Networks and Deep Learning

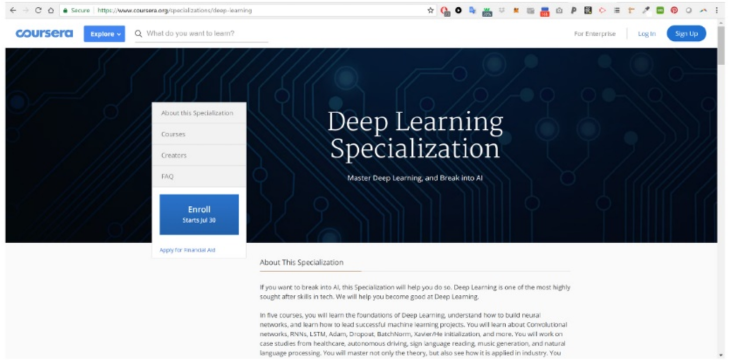
Michael Nielsen

Online Resources



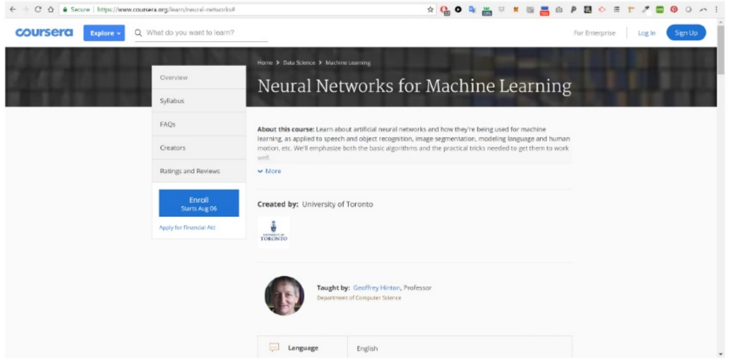
Deep Learning Book

Ian Goodfellow, Yoshua Bengio, Aaron Courville



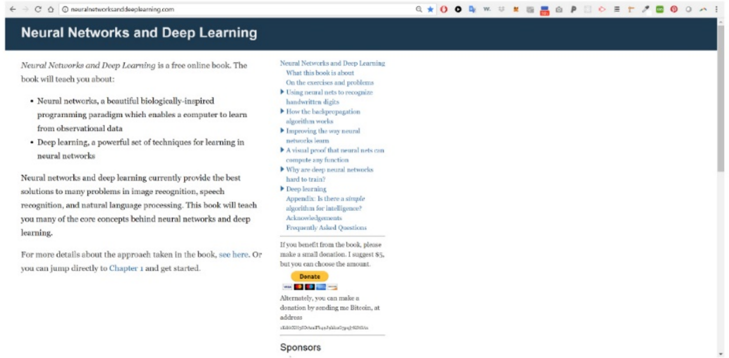
Deep Learning Specialization

Andrew Ng @ Coursera



Neural Networks for Machine Learning

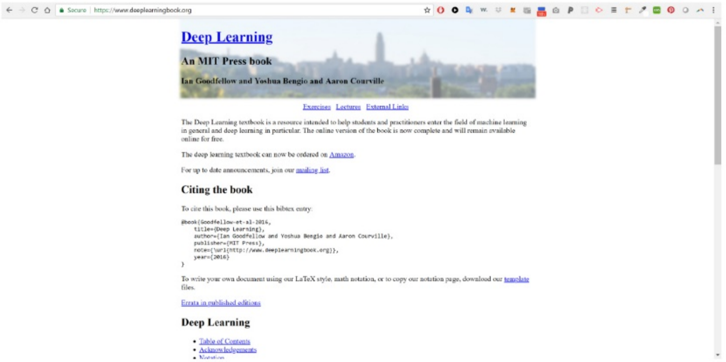
Geoffrey Hinton @ Coursera



Neural Networks and Deep Learning

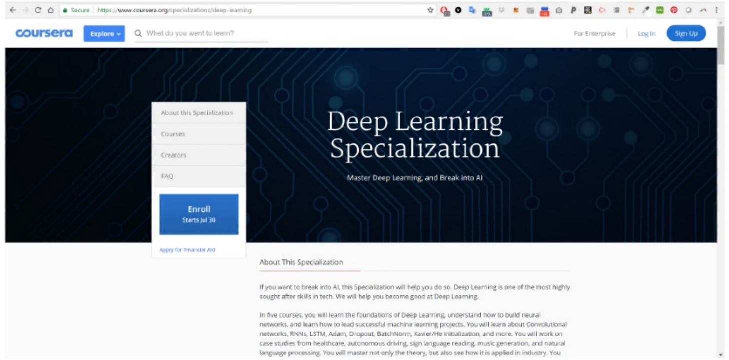
Michael Nielsen

Online Resources



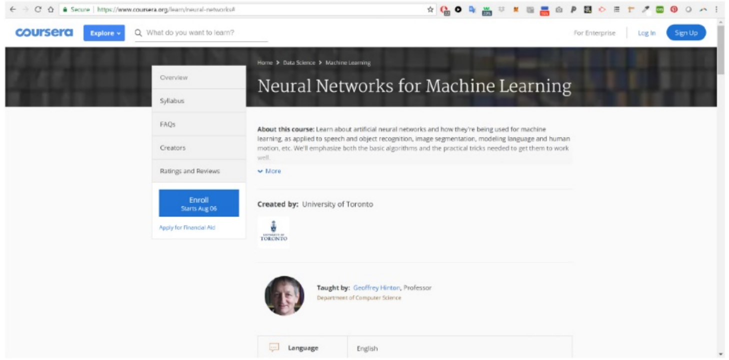
Deep Learning Book

Ian Goodfellow, Yoshua Bengio, Aaron Courville



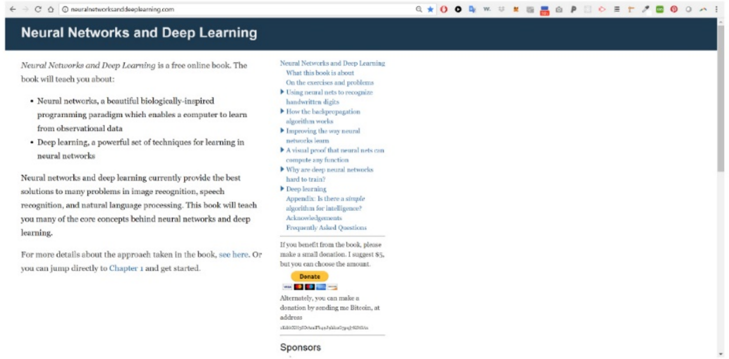
Deep Learning Specialization

Andrew Ng @ Coursera



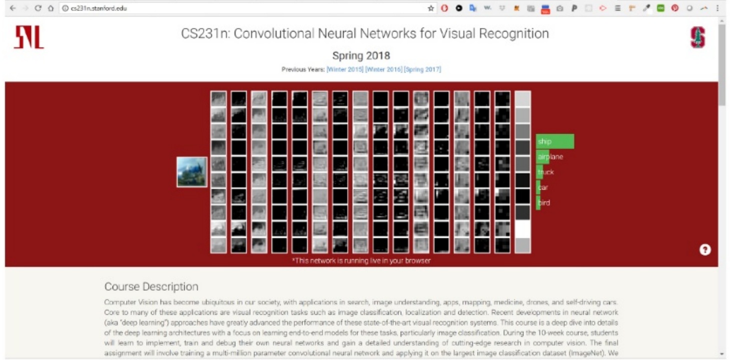
Neural Networks for Machine Learning

Geoffrey Hinton @ Coursera



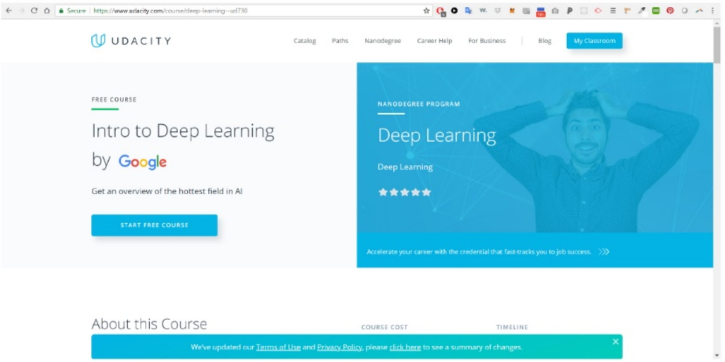
Neural Networks and Deep Learning

Michael Nielsen



CS231n: CNNs for Visual Recognition

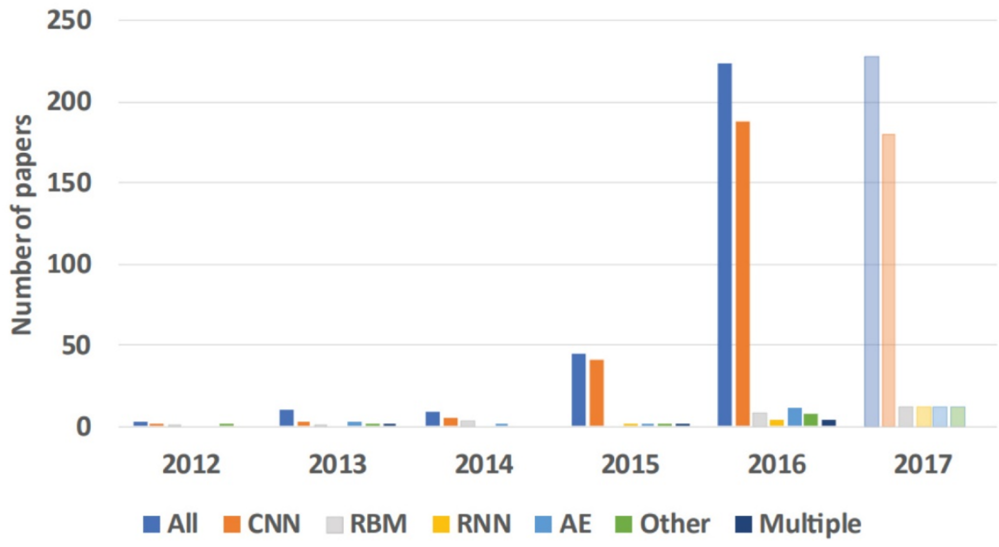
Misc.



Intro to Deep Learning

Udacity

Reproducibility



Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, et al.

A Survey on Deep Learning in Medical Image Analysis
Medical Image Analysis - 2017

Misc.

Open-Source Deep Learning Tools
github.com

Deep Learning

Keras
DL4J
Caffe2
PYTORCH
torch
Caffe
Microsoft CNTK
TensorFlow
theano
mxnet

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, et al.

A Survey on Deep Learning in Medical Image Analysis
Medical Image Analysis - 2017

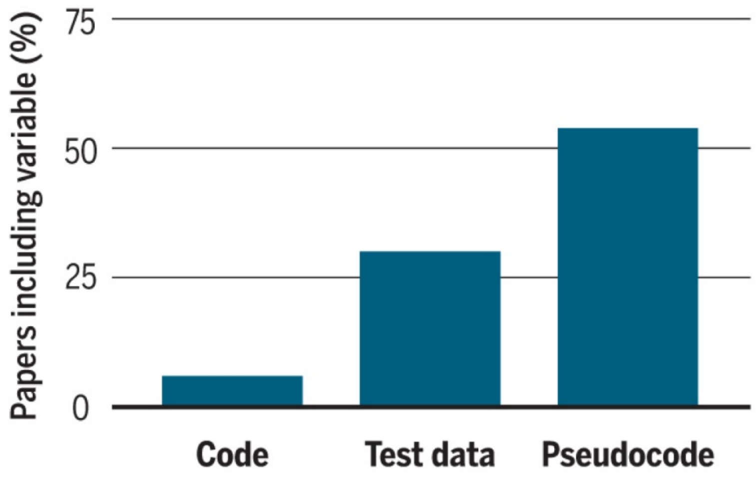
Misc.

Open-Source Deep Learning Tools
github.com

Reproducibility

Code break

In a survey of 400 artificial intelligence papers presented at major conferences, just 6% included code for the papers' algorithms. Some 30% included test data, whereas 54% included pseudocode, a limited summary of an algorithm.



Matthew Hutson

Artificial Intelligence Faces Reproducibility Crisis

Science - 2018

Existing Solutions

houseroad Rename ZFNet to ZFNet-512 (#36)		Latest commit 3be4824 11 hours ago
bvlc_alexnet	Update bvlc_alexnet model	4 months ago
bvlc_googlenet	Add the value_info.json for the remaining of the models except style ...	3 months ago
bvlc_reference_caffenet	Add the value_info.json for the remaining of the models except style ...	3 months ago
bvlc_reference_rcnn_ilsvrc13	Add the value_info.json for the remaining of the models except style ...	3 months ago
densenet121	Add DenseNet-121 model	4 months ago
detectron	Add Detectron e2e_faster_rcnn_R-50-C4_2x model	3 months ago
inception_v1	Add Inception models	4 months ago
inception_v2	Add Inception models	4 months ago
resnet50	Add ResNet-50 model	4 months ago
scripts	Add Detectron e2e_faster_rcnn_R-50-C4_2x model	3 months ago
squeezenet	Correct SqueezeNet value_info to 227x227	3 months ago
style_transfer	Add other style transfer models	4 months ago
vgg19	Add VGG models	4 months ago
zfnet512	Rename ZFNet to ZFNet-512 (#36)	11 hours ago
.gitattributes	Remove squeezenet-specific lines from .gitattributes.	4 months ago
LICENSE	Add Apache 2.0 license	4 months ago
README.md	Update README to describe subdirectory access	3 months ago

Yangqing Jia, Evan Shelhamer, Jeff Donahue, et al.

Caffe: Convolutional Architecture for Fast Feature Embedding

arxiv.org/abs/1408.5093

Samim and Graphific

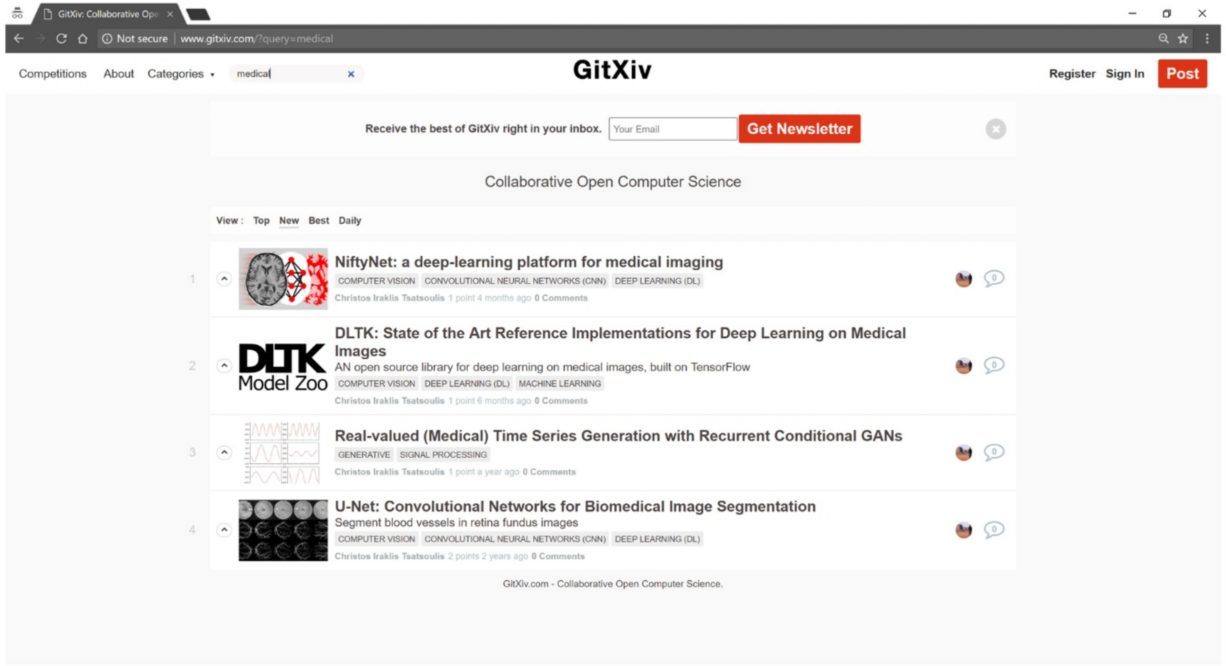
GitXiv—Collaborative Open Computer Science

gitxiv.com

Existing Solutions

Yangqing Jia, Evan Shelhamer, Jeff Donahue, et al.

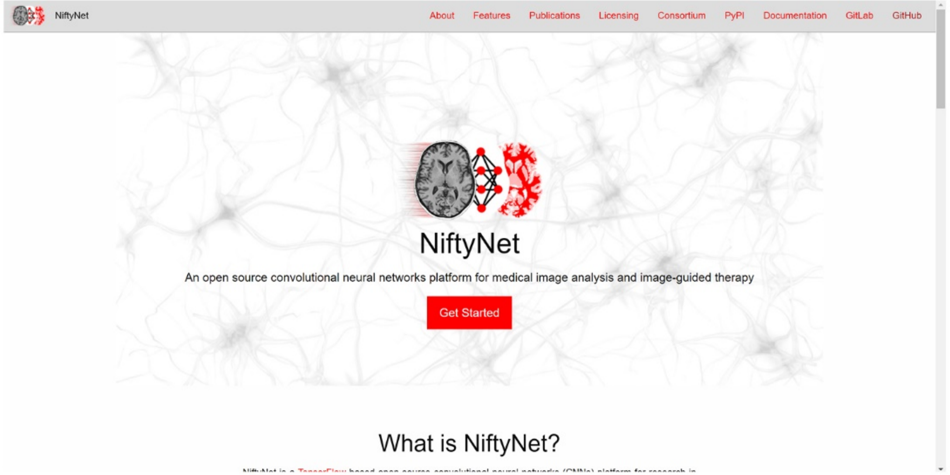
Caffe: Convolutional Architecture for Fast Feature Embedding
arxiv.org/abs/1408.5093

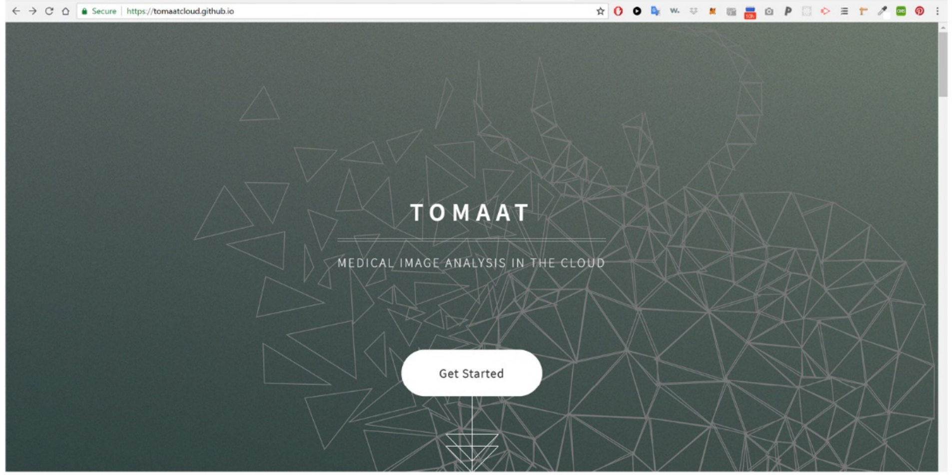


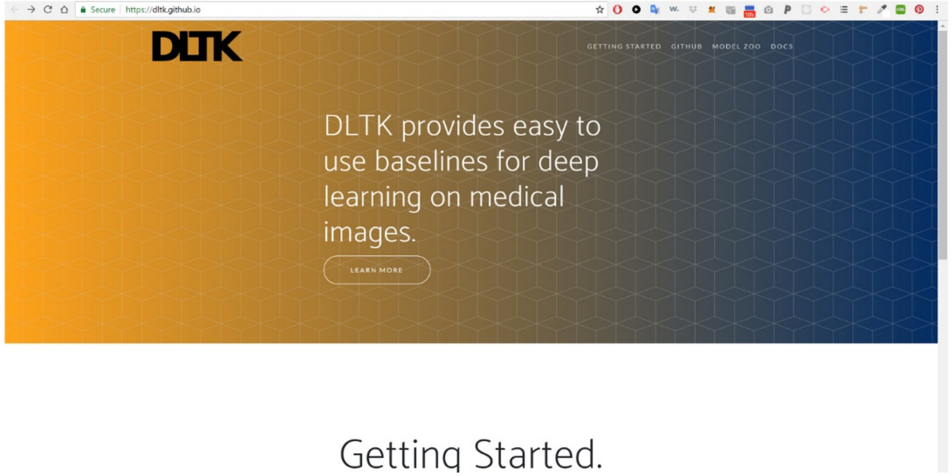
Samim and Graphific

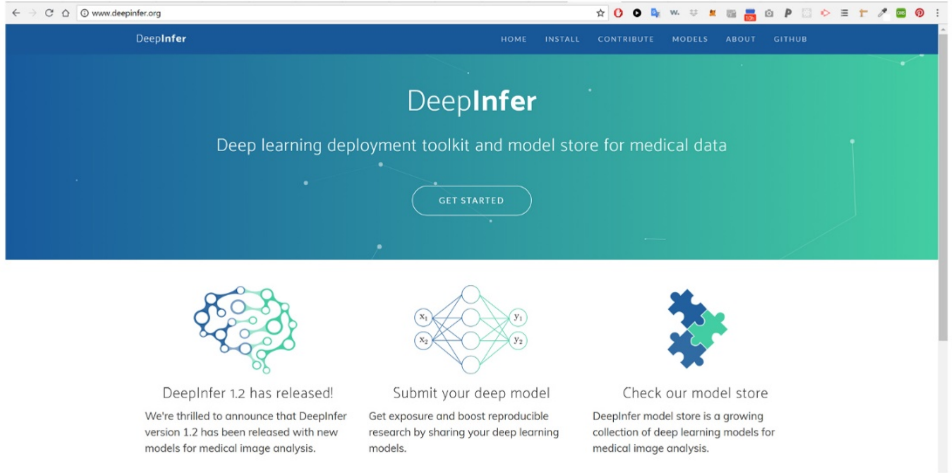
GitXiv—Collaborative Open Computer Science
gitxiv.com

Existing Medical Imaging Solutions





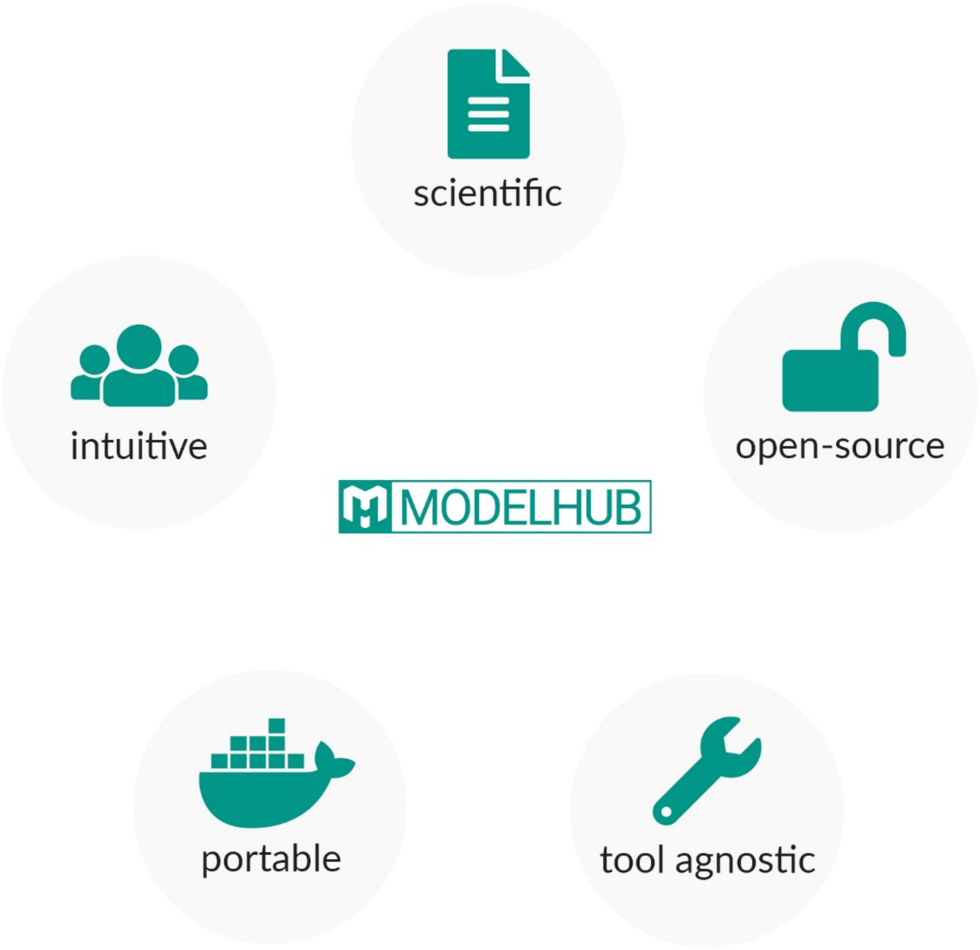




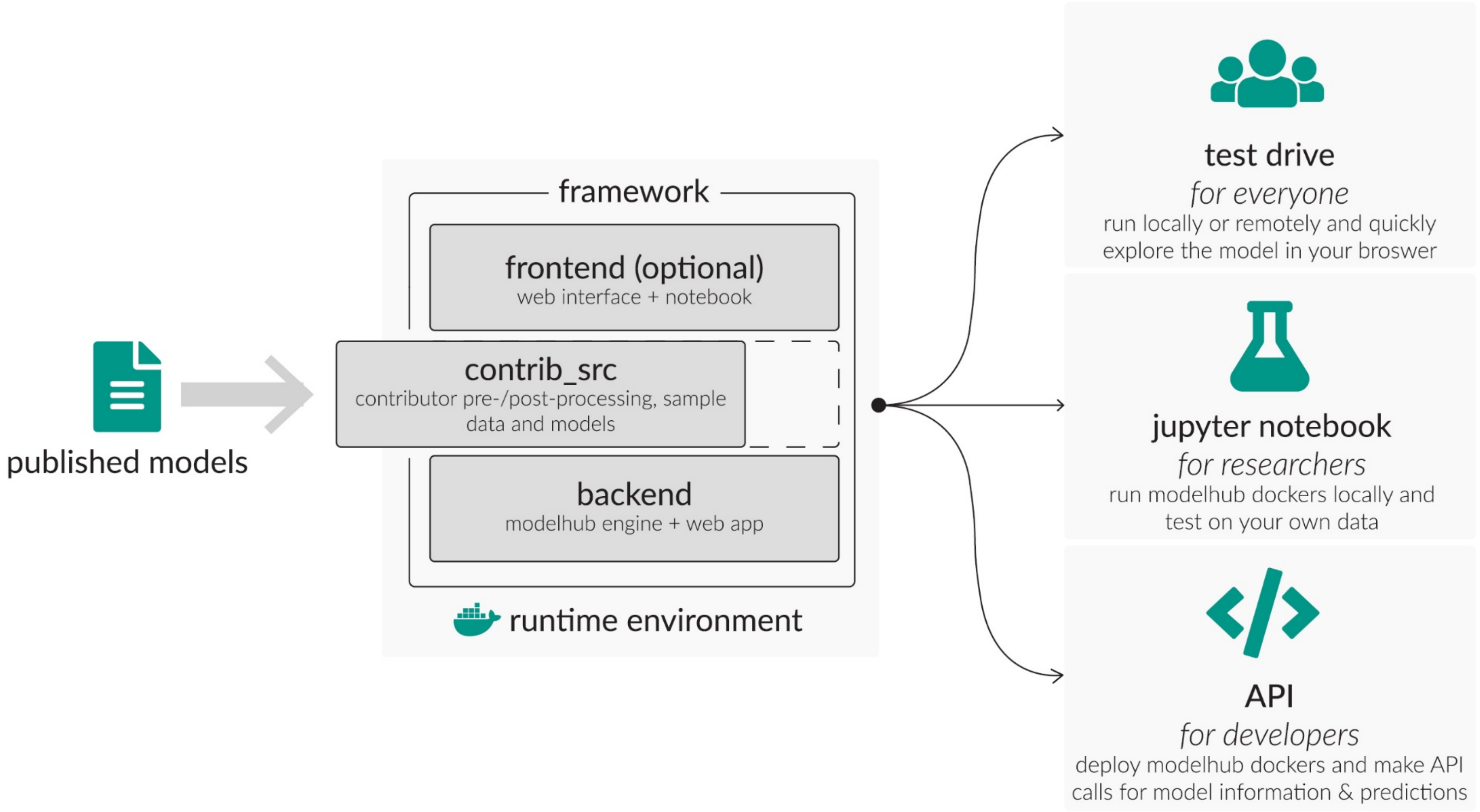
MODELHUB

www.modelhub.ai

Components

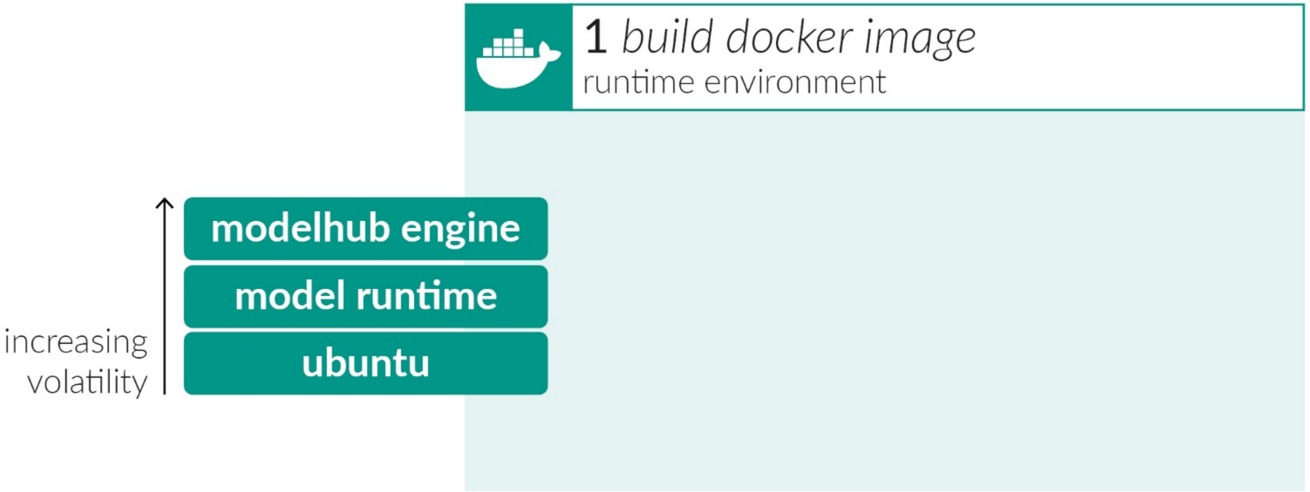

scientific
intuitive
open-source
MODELHUB
portable
tool agnostic

How it Works

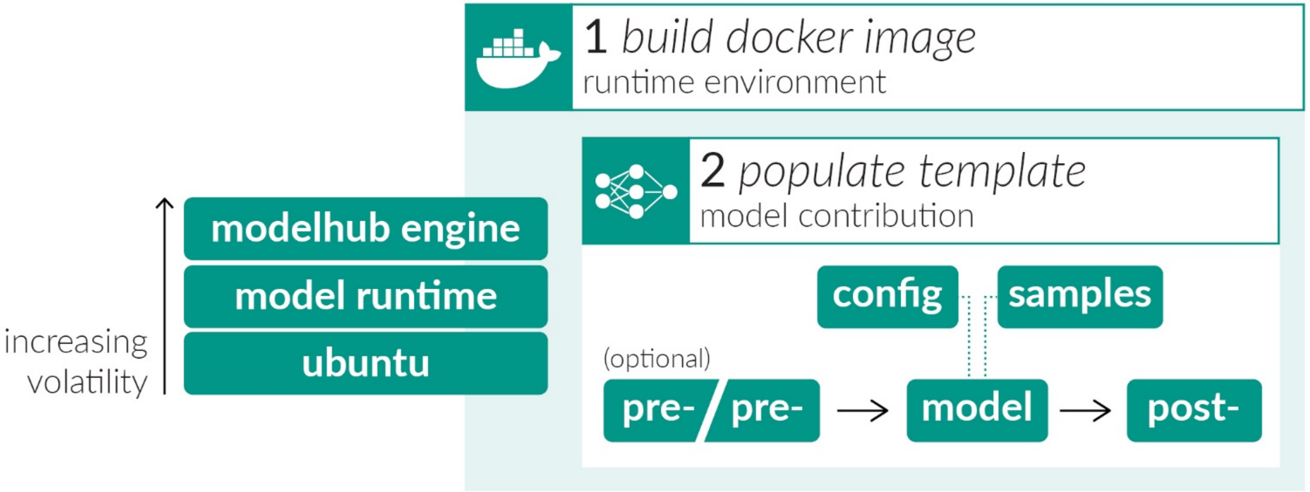


published models
framework
frontend (optional)
web interface + notebook
contrib_src
contributor pre-/post-processing, sample data and models
backend
modelhub engine + web app
runtime environment
test drive
for everyone
run locally or remotely and quickly explore the model in your broswer
jupyter notebook
for researchers
run modelhub dockers locally and test on your own data
API
for developers
deploy modelhub dockers and make API calls for model information & predictions

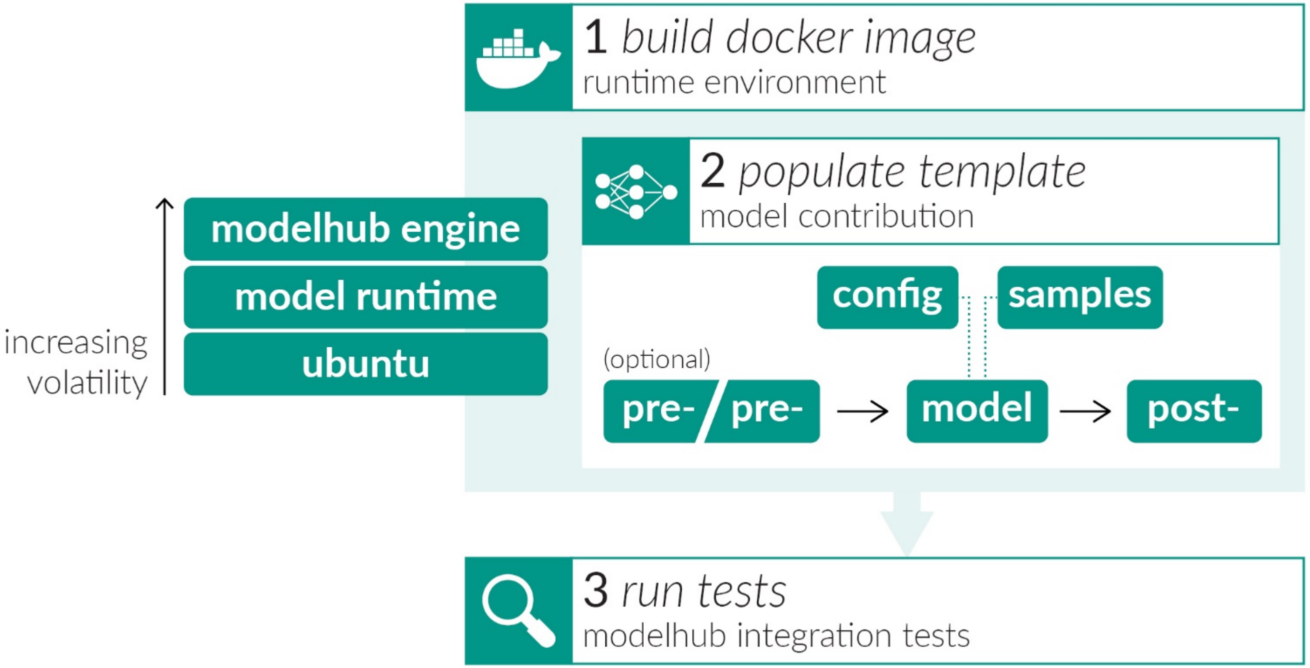
For Contributors



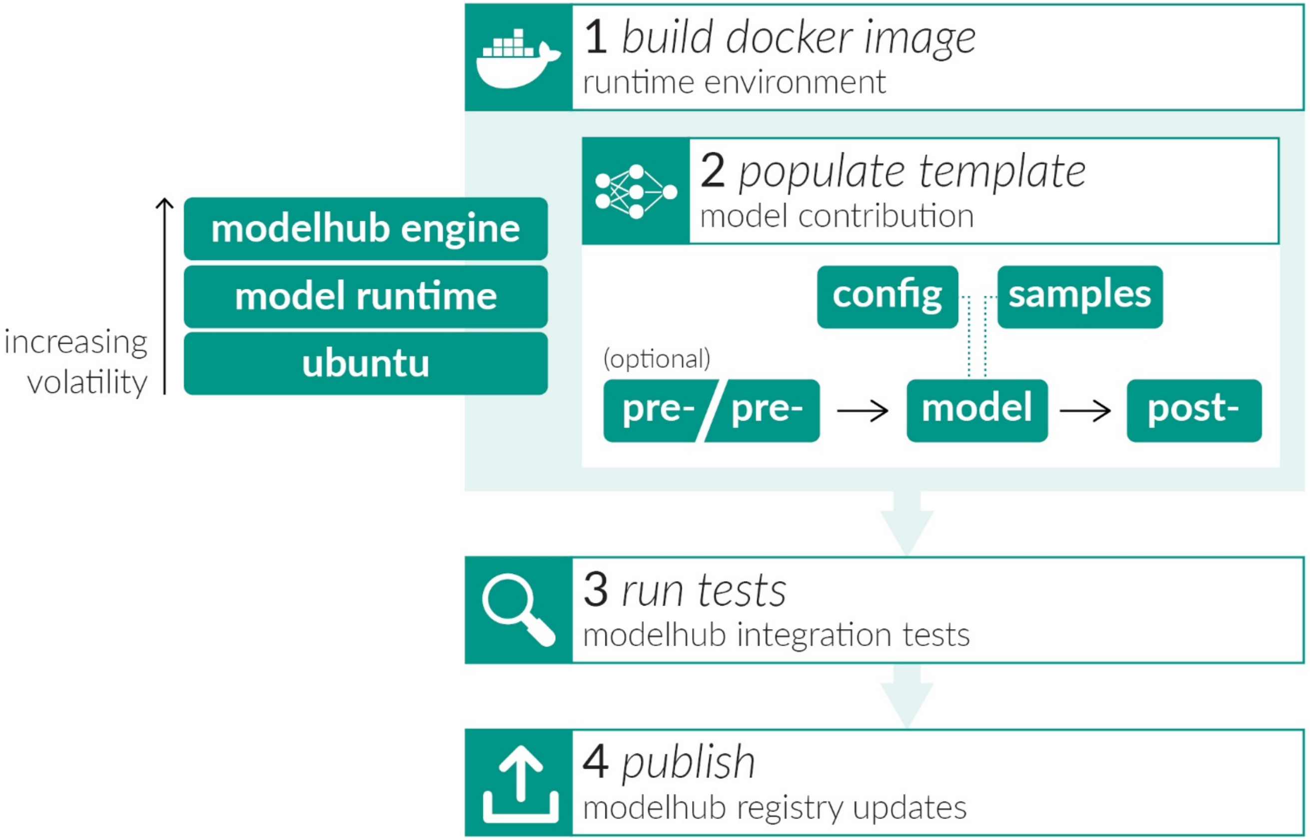
For Contributors



For Contributors



For Contributors



deep learning models for

pathology.

COLLECTION

Community Outreach

MODELHUB

info@modelhub.ai

co-authorship through model contributions

Thank you!