Road Segmentation in Satellite Imagery

Niklas Klein and Jann Goschenhofer

March 29, 2018

Structure

Structure

Statistical Consulting

- Ingredient of the MSc in Statistics at LMU
- Task: process a data driven project in cooperation with an industrial partner
- Workload: 12 ECTS

Structure

Problem

- Our industrial partner: a reinsurance company with headquarter in munich
- One of their current projects includes Latin American road systems

Problem

- Our industrial partner: a reinsurance company with headquarter in munich
- One of their current projects includes Latin American road systems
- Problems with wrong, incomplete or outdated data:

Problem

- Our industrial partner: a reinsurance company with headquarter in munich
- One of their current projects includes Latin American road systems
- Problems with wrong, incomplete or outdated data:

Goals I/II

- **Improve data quality of current project in Latin America**
- Monitor the development of infrastructure construction projects (via drones, airplanes, ...)
- Track changes in infrastructure after certain events (earthquakes, tsunamis, ...)
- Extension: distinguish different road types

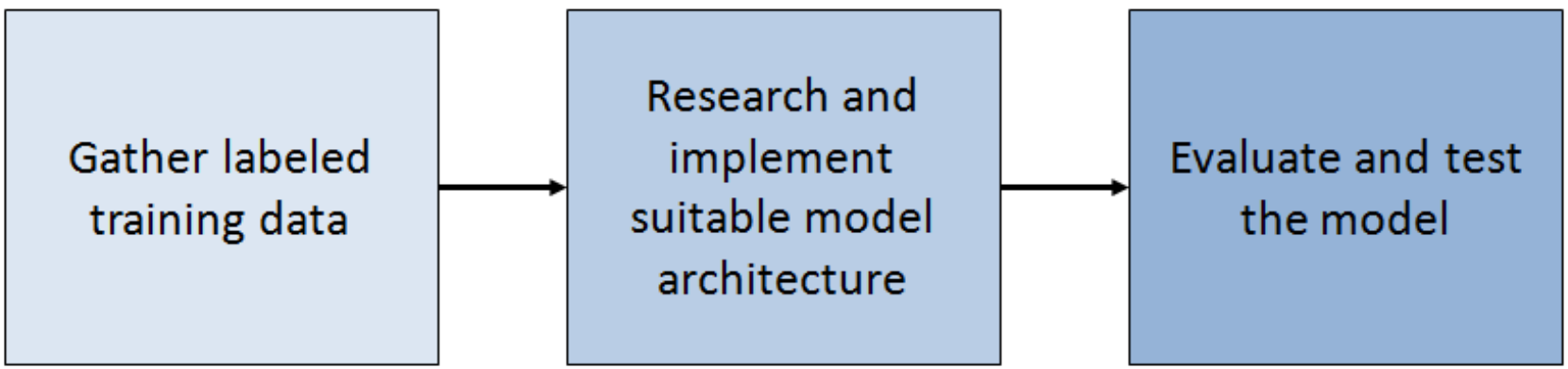
$\Rightarrow$ Focus on *Proof of Concept* during our consulting project to establish a basis for further development

Goals II/II

- What might be a suitable model?
- Hardware requirements/limitations
- Research Questions
  1. Influence of environmental landscape
  2. Influence of zoom
  3. Required train data quality
  4. Stability of the model
  5. Differencies between rural and urban landscape
  6. Required amount of train data
  7. General difficulties

Approach

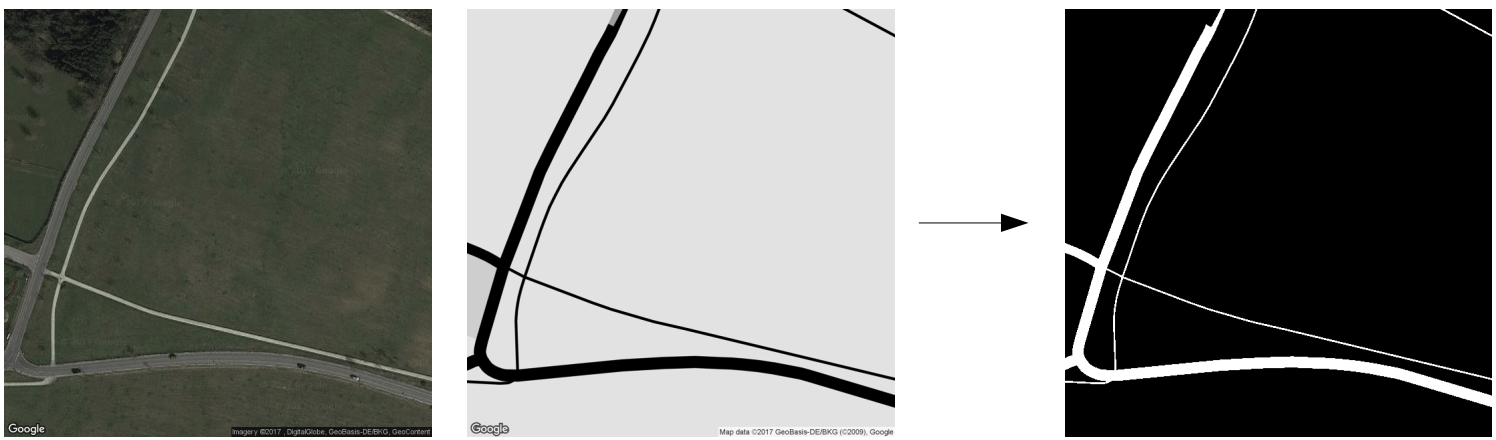
- Task: create and train a model that detects if pixel (i,j) of the satellite image correponds to a road or not
- Scientific problem definition: **pixelwise image segmentation**
- Popular model class for such problems: fully convolutional neural networks (FCN)



Structure

Train data gathering and automated labeling

- Typical ML-problem: how to get labeled train data?
- Idea: Google Maps Static API offers https-based retrieval of maps data
- API accepts a variety of input parameters to retrieve pairs of inputs and labels for given coordinate:



$\Rightarrow$ generate binary label mask for each satellite image!

Structure

Filters and the Convolution

- How to represent a digital image?

Filters and the Convolution

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

- Basically as an array of integers

Filters and the Convolution

Sobel-Operator

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \boxed{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

- The Sobel-Operator computes an approximation of the gradient of the image intensity function.
- G_x enables us to to detect horizontal edges!

Filters and the Convolution

Sobel-Operator

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \boxed{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

Filters and the Convolution

Sobel-Operator

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \boxed{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

0	0	0	0	255	255	0	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	0	255	255	0	0	0	0
255	0	0	255	255	255	255	0	0	0
0	0	255	255	255	255	255	255	0	0
0	255	0	255	255	255	255	0	255	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	255	255	255	0	0	0
0	0	0	255	0	0	255	0	0	255
0	0	255	255	0	0	255	255	0	0

$$\begin{aligned} S_{(i,j)} = (I \star G_x)_{(i,j)} = & -1 \cdot 0 + 0 \cdot 255 + \mathbf{1 \cdot 255} \\ & -2 \cdot 0 + 0 \cdot 0 + \mathbf{2 \cdot 255} \\ & -1 \cdot 0 + 0 \cdot 255 + \mathbf{1 \cdot 255} \end{aligned}$$

Filters and the Convolution

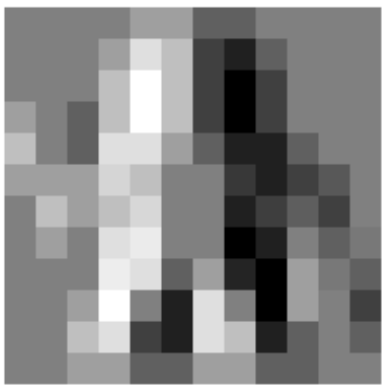
1020

Filters and the Convolution

0	0	0	0	255	255	-255	-255	0	0	0	0
0	0	0	255	765	510	-510	-765	-255	0	0	0
0	0	0	510	1020	510	-510	-1020	-510	0	0	0
255	0	-255	510	1020	510	-510	-1020	-510	0	0	0
510	0	-255	765	765	255	-255	-765	-765	-255	0	0
255	255	255	765	510	0	0	-510	-765	-510	-255	0
0	510	255	510	765	0	0	-765	-510	-255	-510	0
0	255	0	765	1020	0	0	-1020	-765	0	-255	0
0	0	0	1020	765	-255	255	-765	-1020	255	0	-255
0	0	255	1020	0	-765	765	0	-1020	255	0	-510
0	0	510	765	-510	-765	765	510	-765	-255	0	-255
0	0	255	255	-255	-255	255	255	-255	-255	0	0

- Applying the Sobel-Operator to every location in the input space yields us the **feature map**.

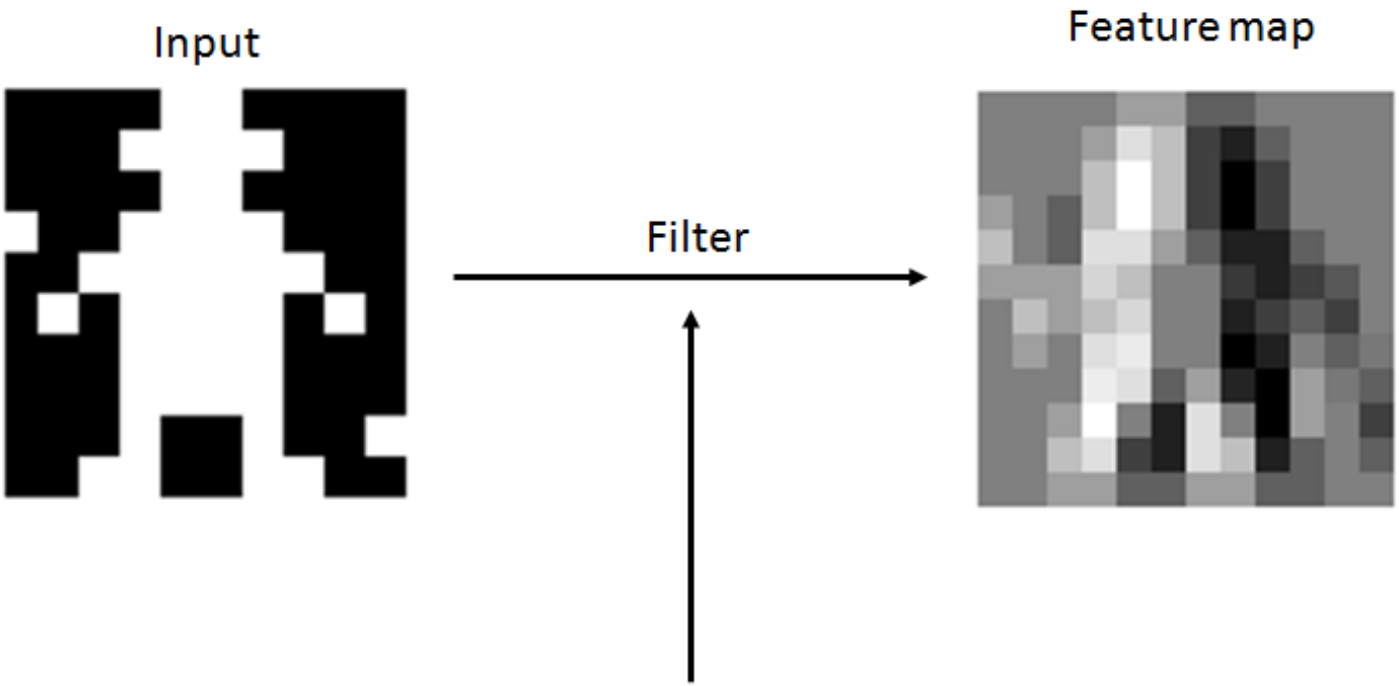
Filters and the Convolution



128	128	128	128	159	159	96	96	128	128	128	128
128	128	128	159	223	191	64	32	96	128	128	128
128	128	128	191	255	191	64	0	64	128	128	128
159	128	96	191	255	191	64	0	64	128	128	128
191	128	96	223	223	159	96	32	32	96	128	128
159	159	159	223	191	128	128	64	32	64	96	128
128	191	159	191	223	128	128	32	64	96	64	128
128	159	128	223	255	128	128	0	32	128	96	128
128	128	128	255	223	96	159	32	0	159	128	96
128	128	159	255	128	32	223	128	0	159	128	64
128	128	191	223	64	32	223	191	32	96	128	96
128	128	159	159	96	96	159	159	96	96	128	128

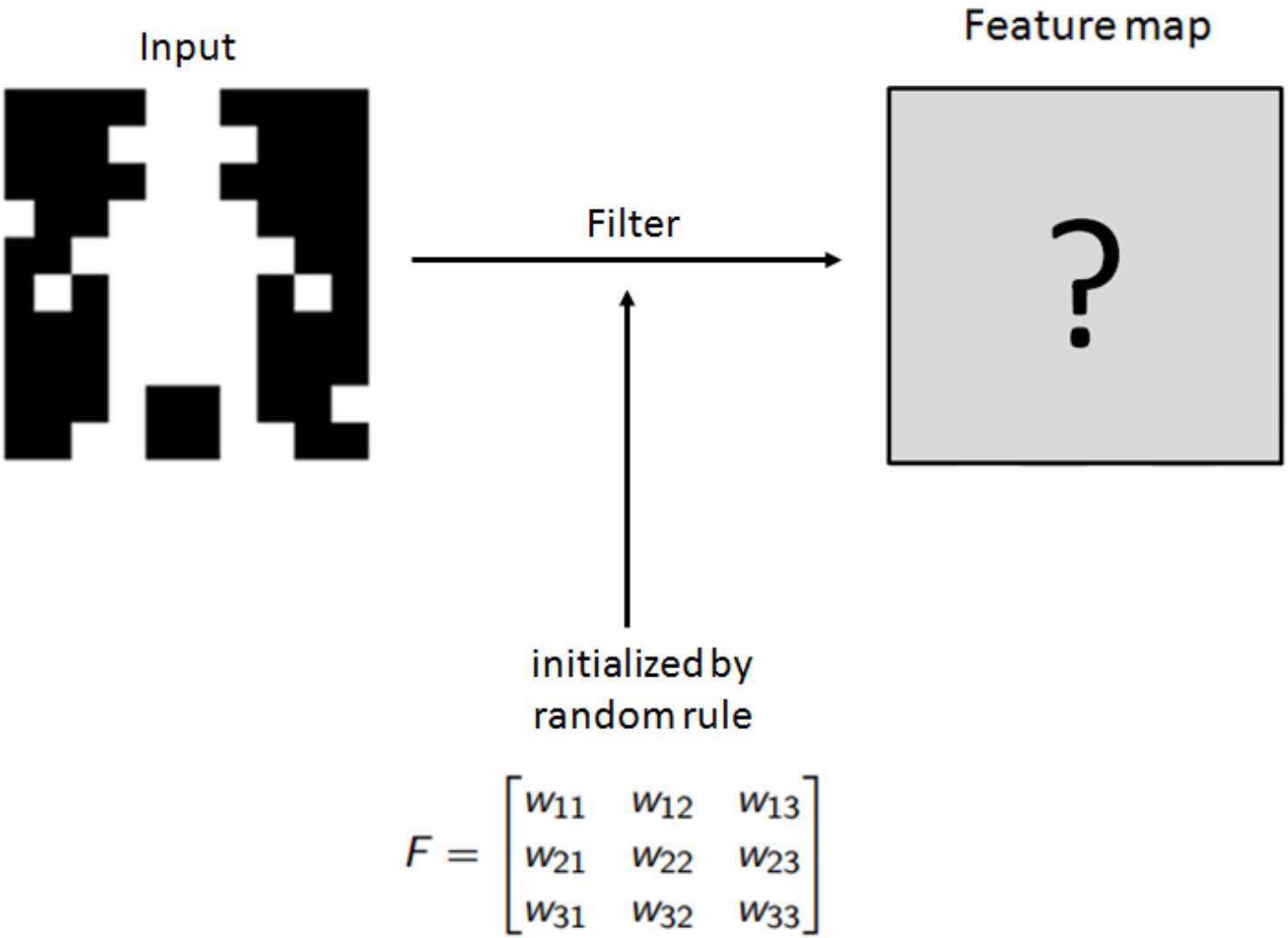
- Normalized feature map reveals horizontal edges.

Filters and the Convolution



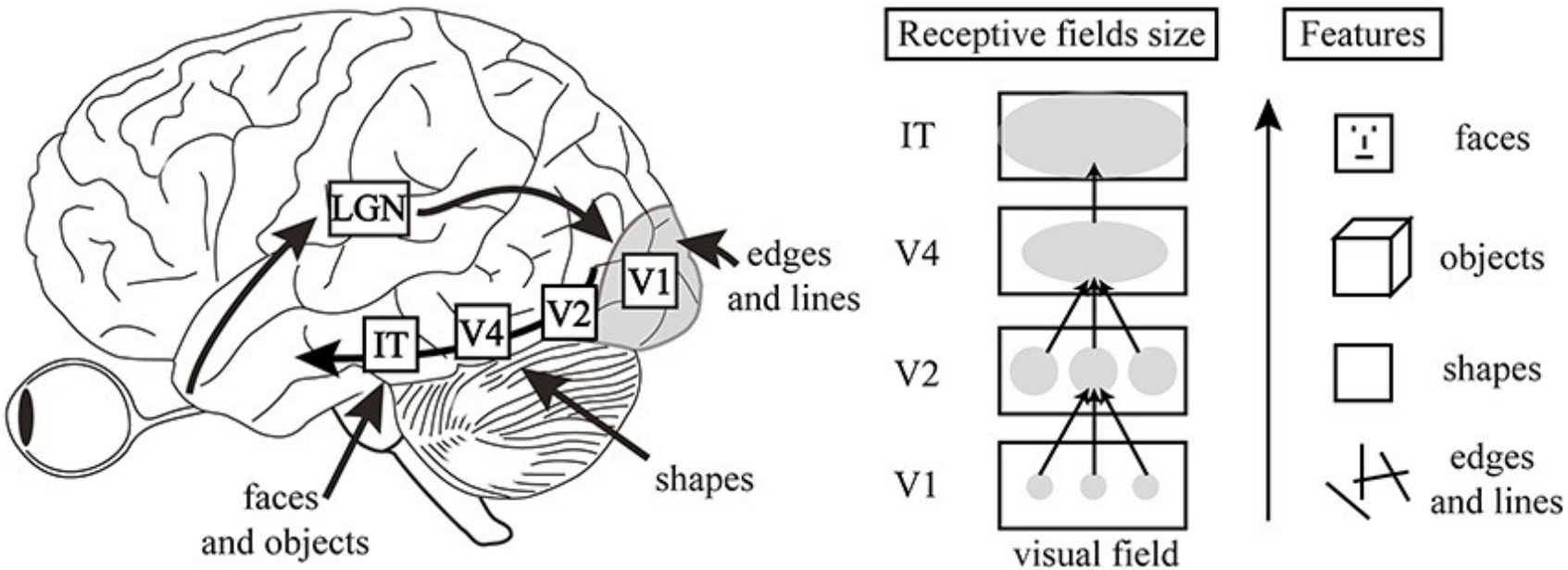
$$F = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

Filters and the Convolution



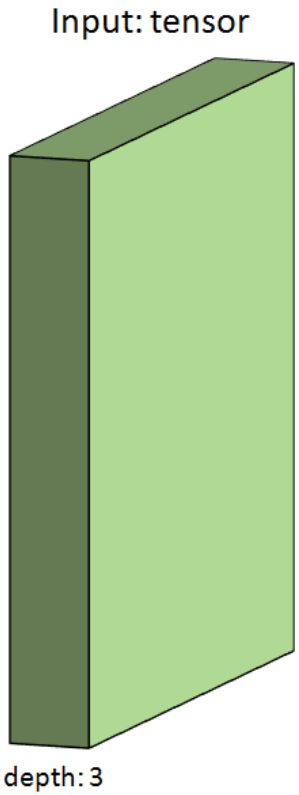
$$F = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

Intuitive Explanation of Convolutional Neural Networks

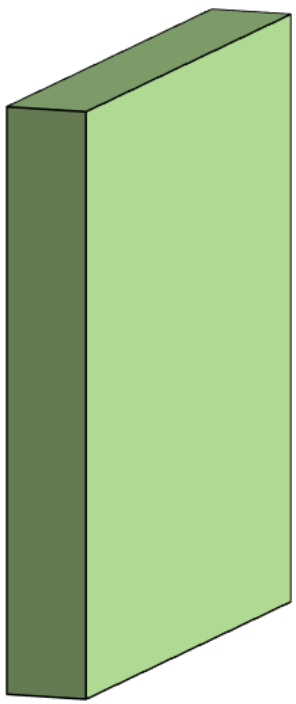


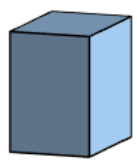
Visual cortex and feature extraction [Herzog & Clarke, 2014].

A Potential Model Architecture

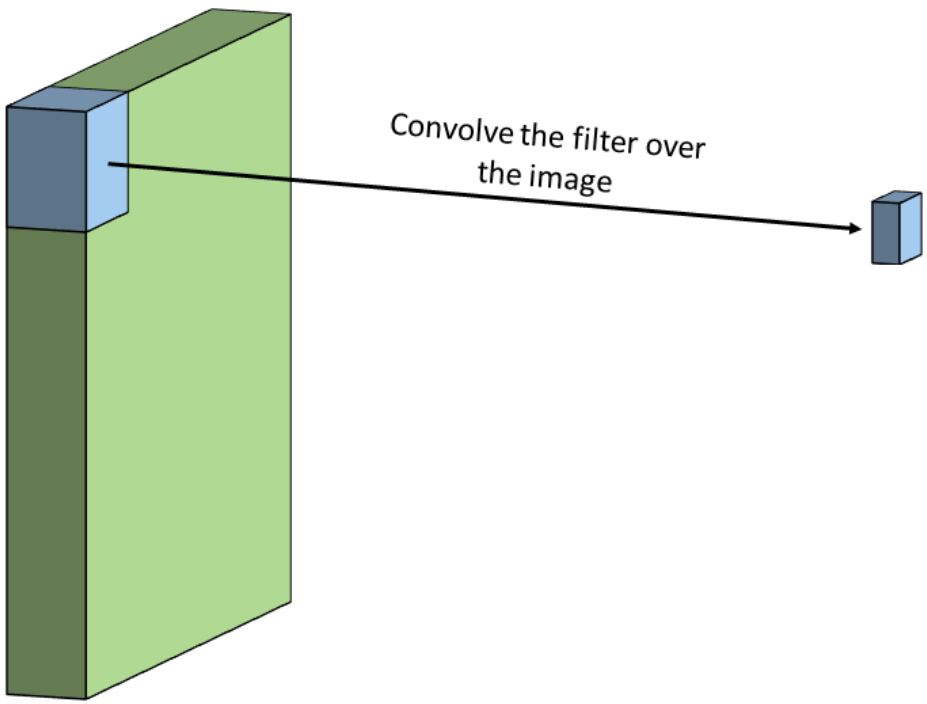


A Potential Model Architecture

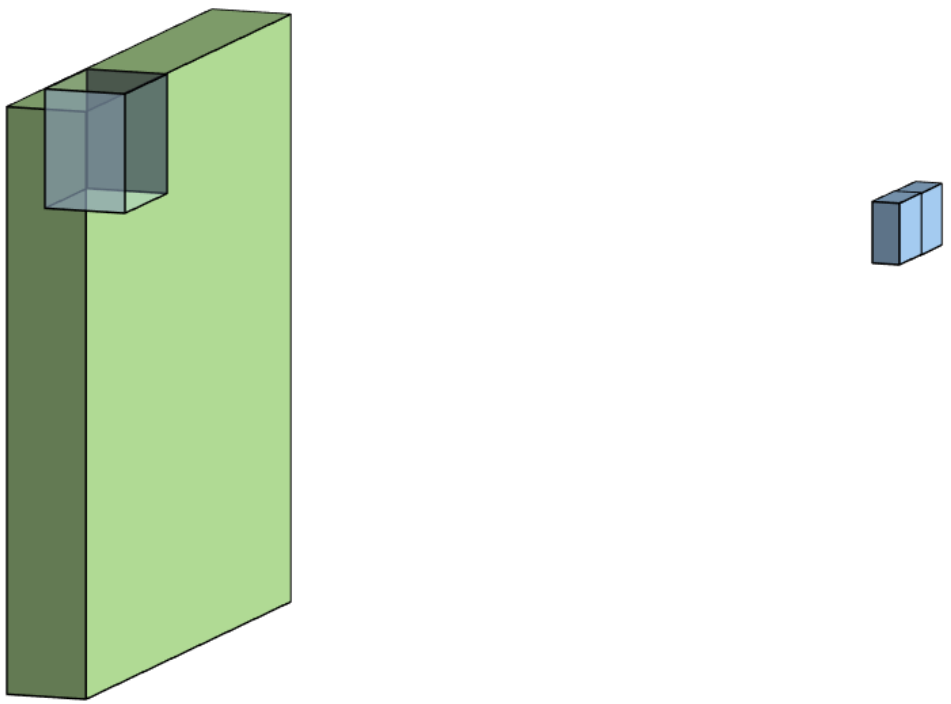




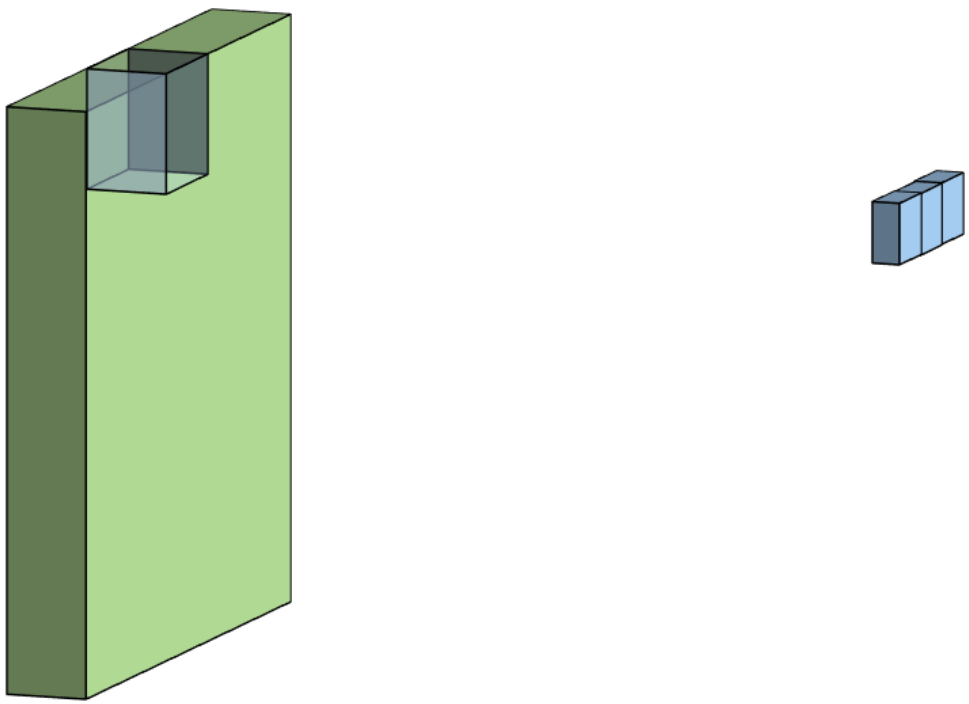
A Potential Model Architecture



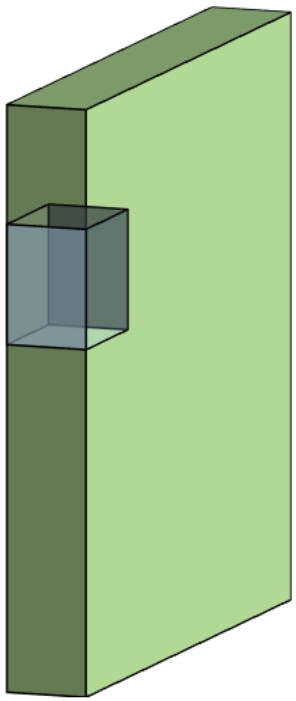
A Potential Model Architecture

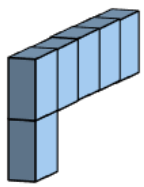


A Potential Model Architecture

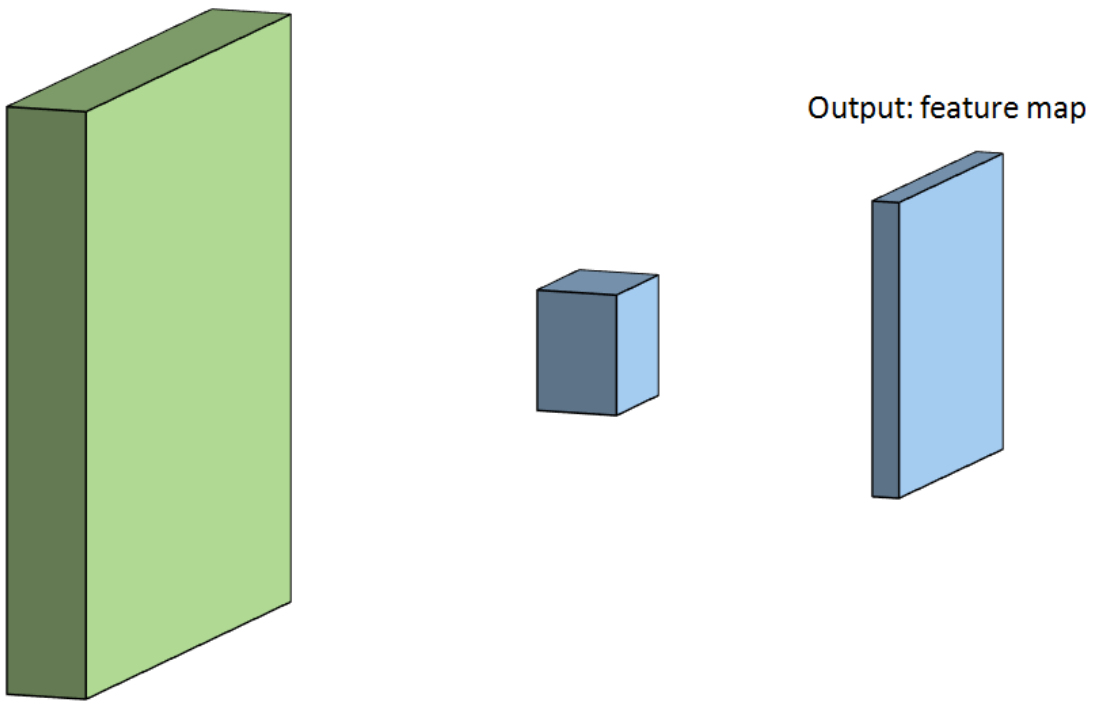


A Potential Model Architecture

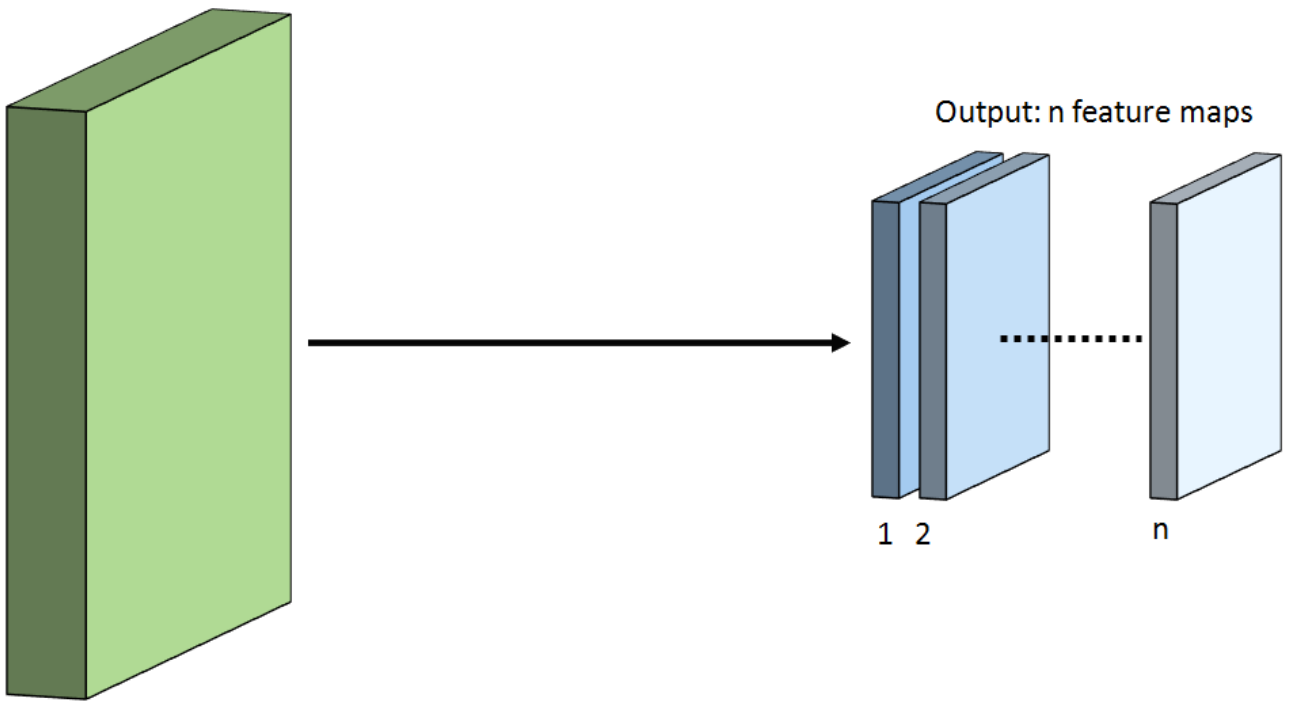




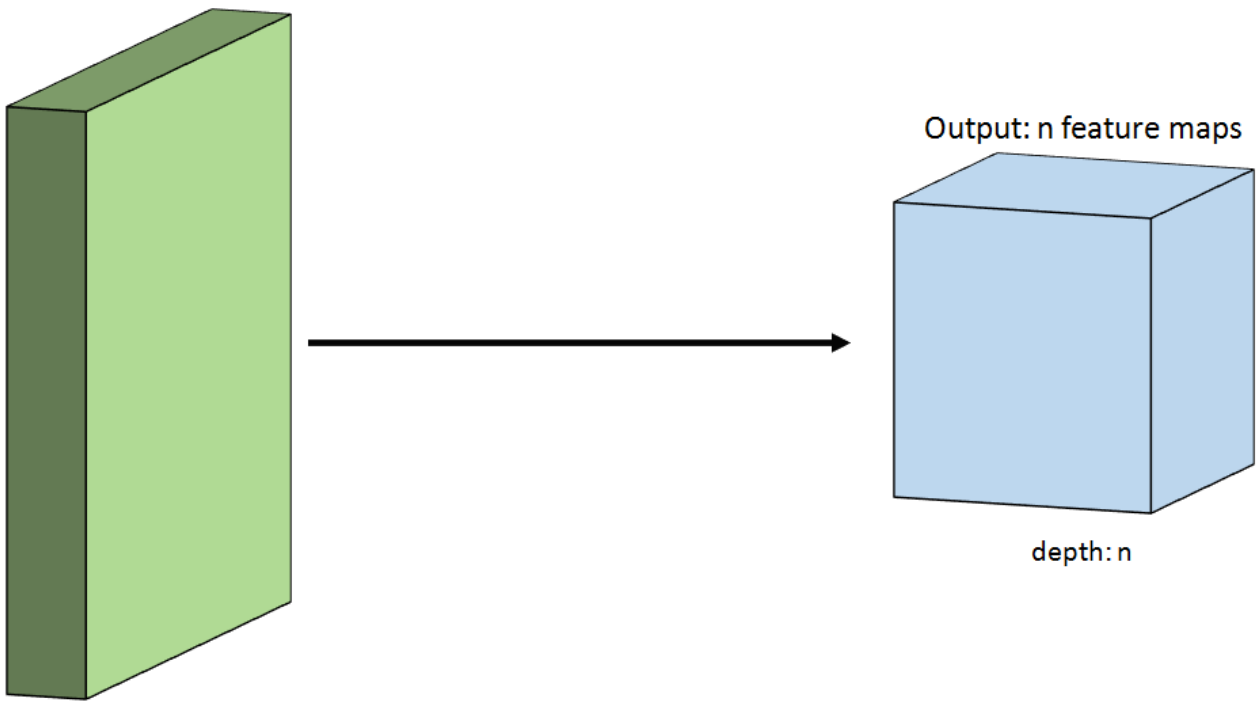
A Potential Model Architecture



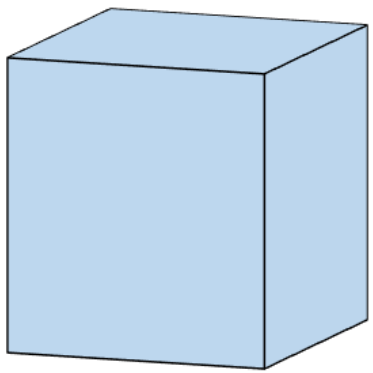
A Potential Model Architecture



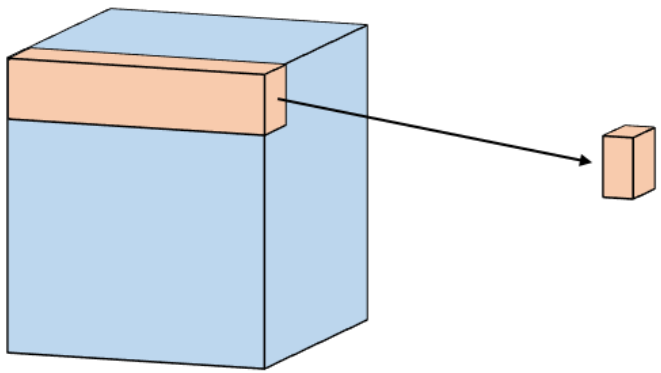
A Potential Model Architecture



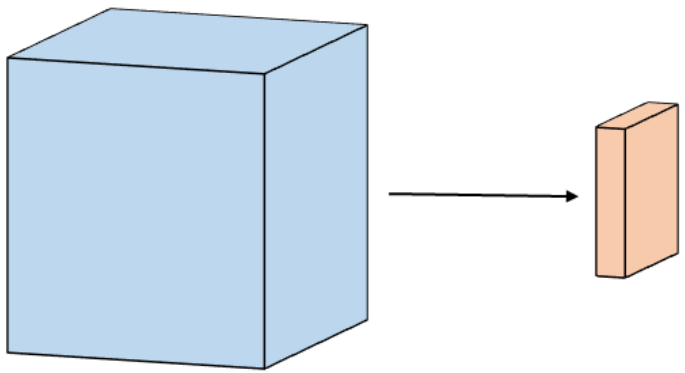
A Potential Model Architecture



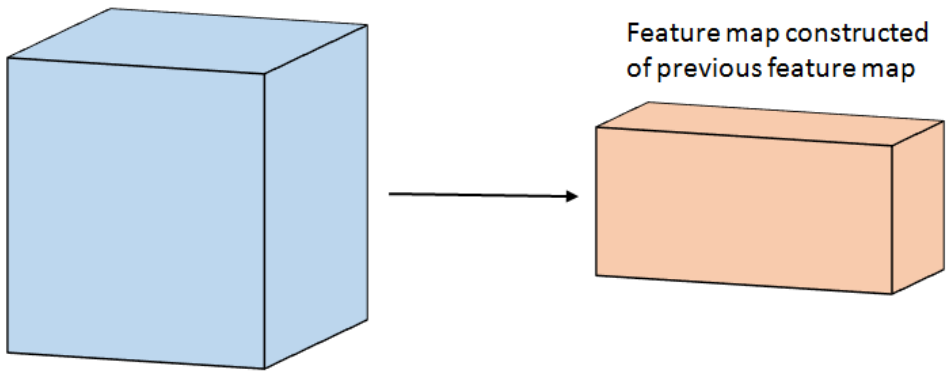
A Potential Model Architecture



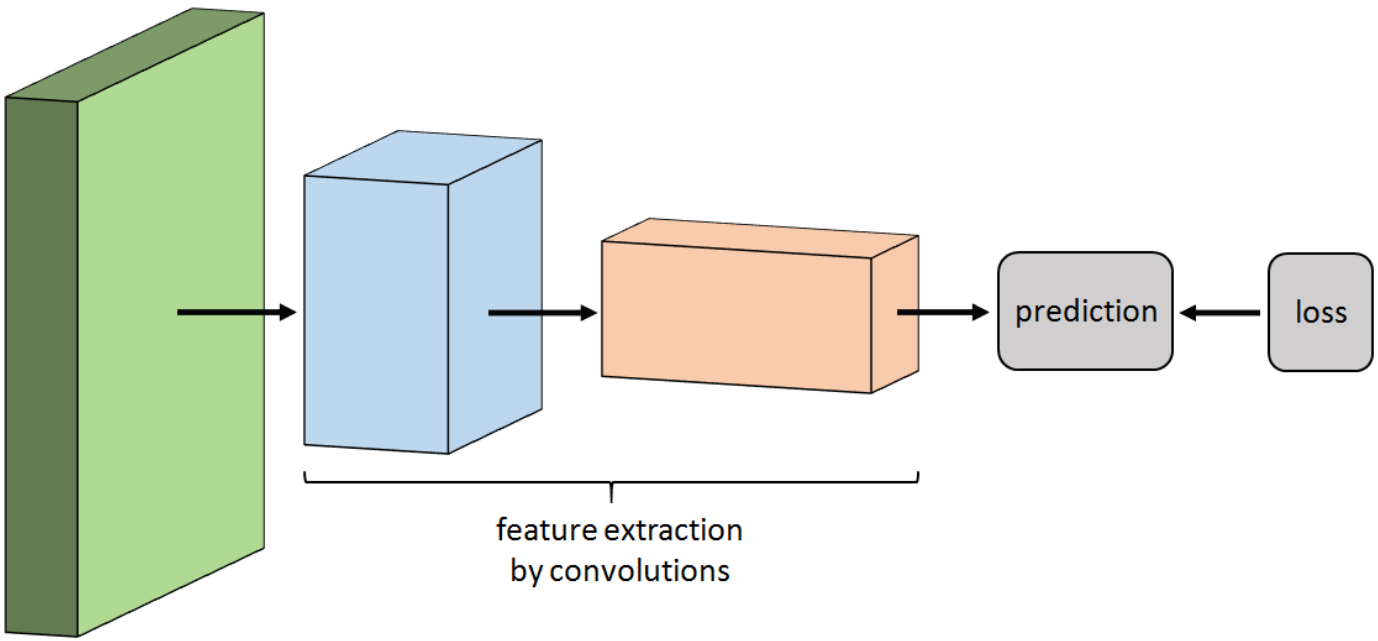
A Potential Model Architecture



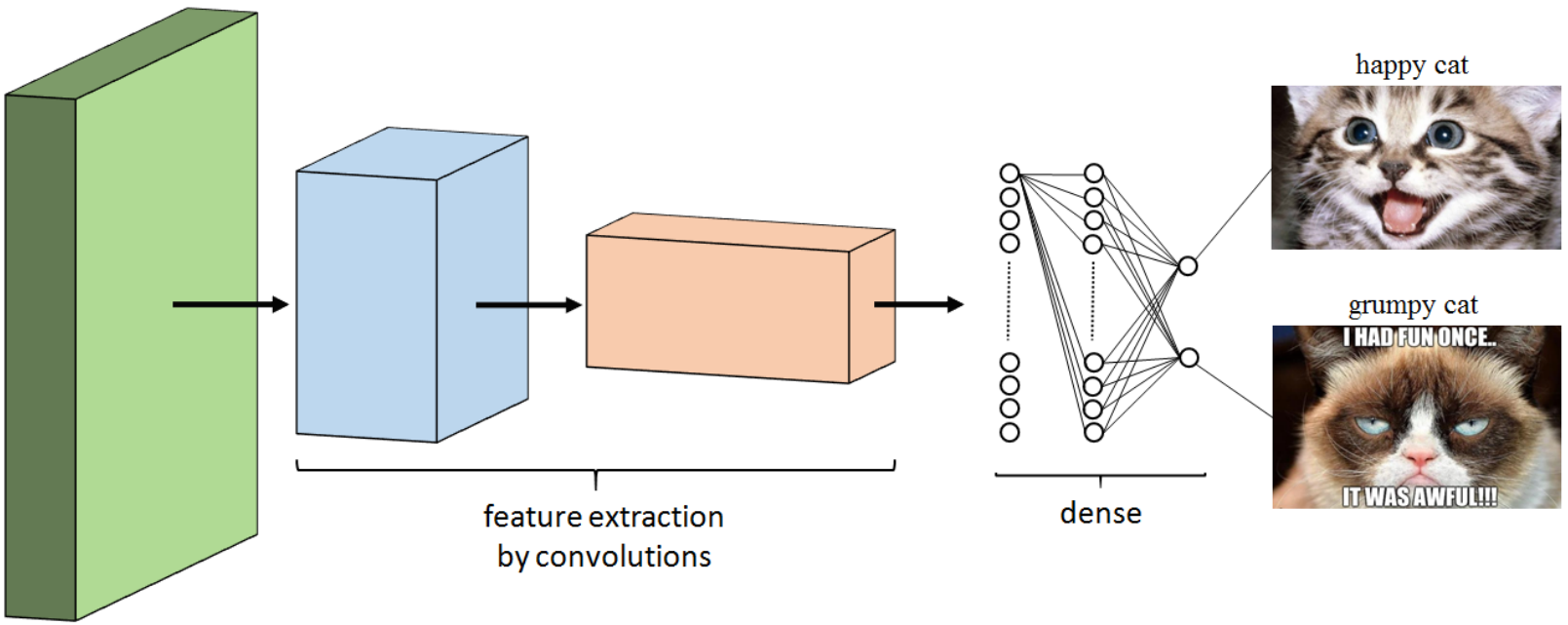
A Potential Model Architecture



A Potential Model Architecture

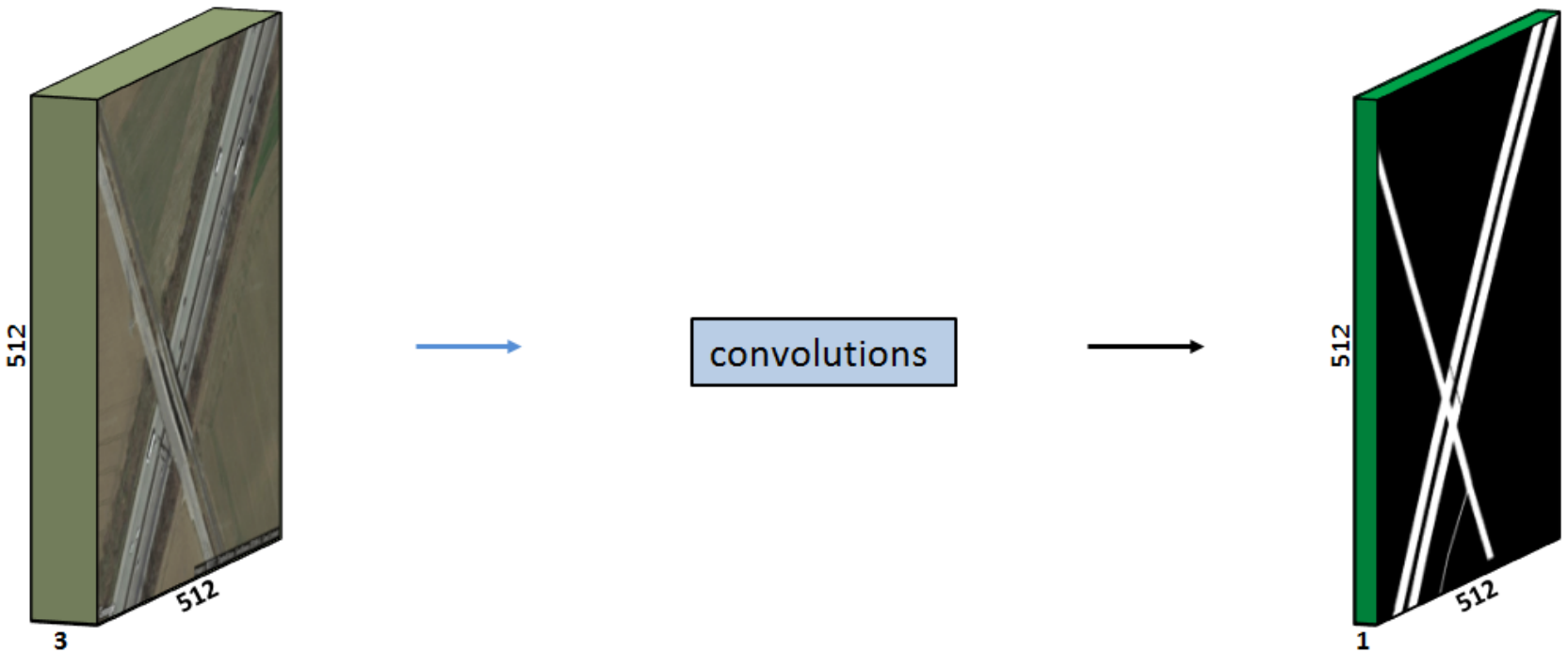


A Potential Model Architecture

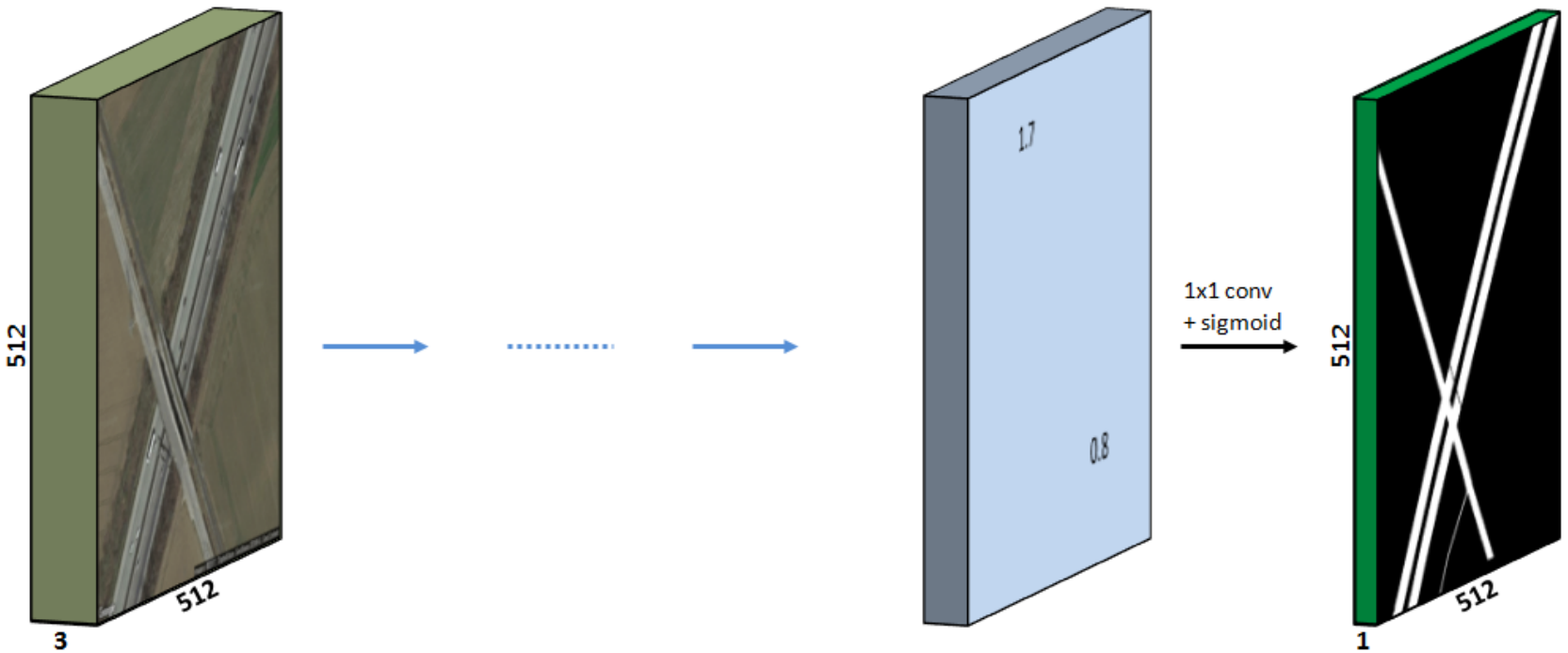


Structure

How to build a suitable architecture

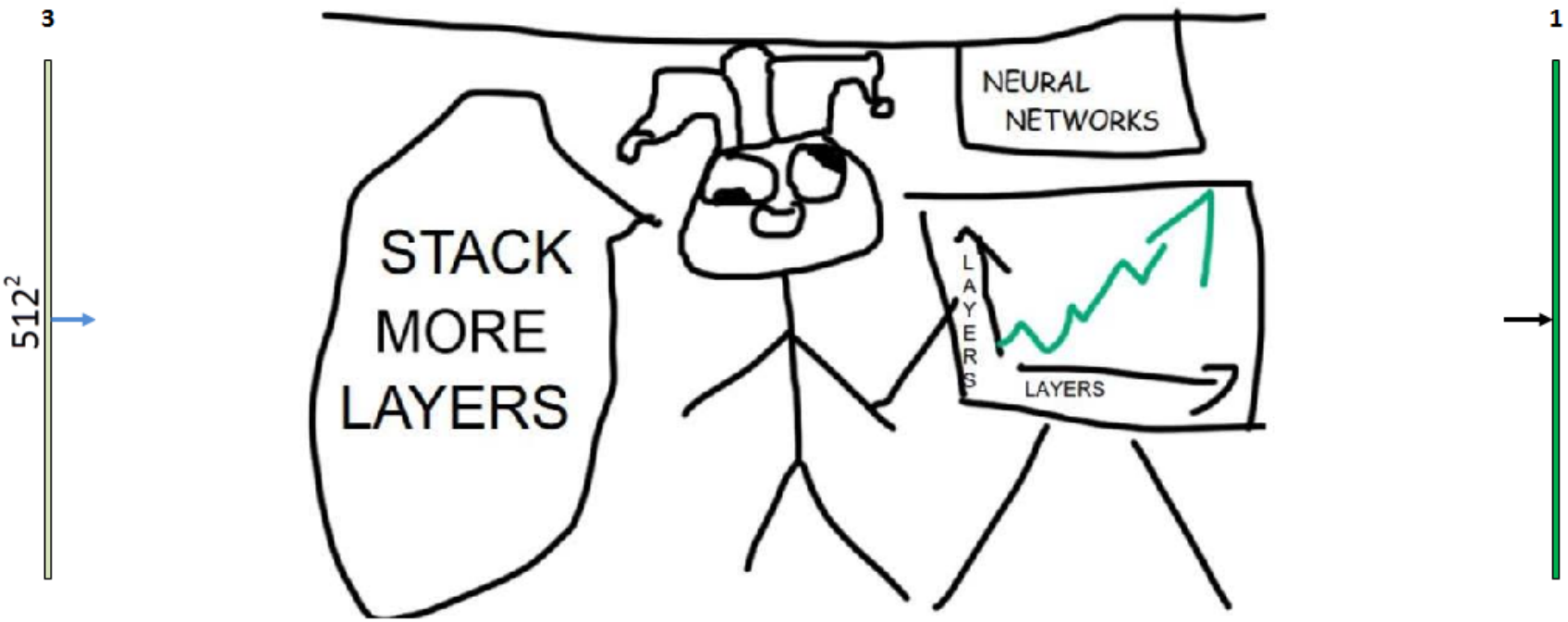


How to build a suitable architecture

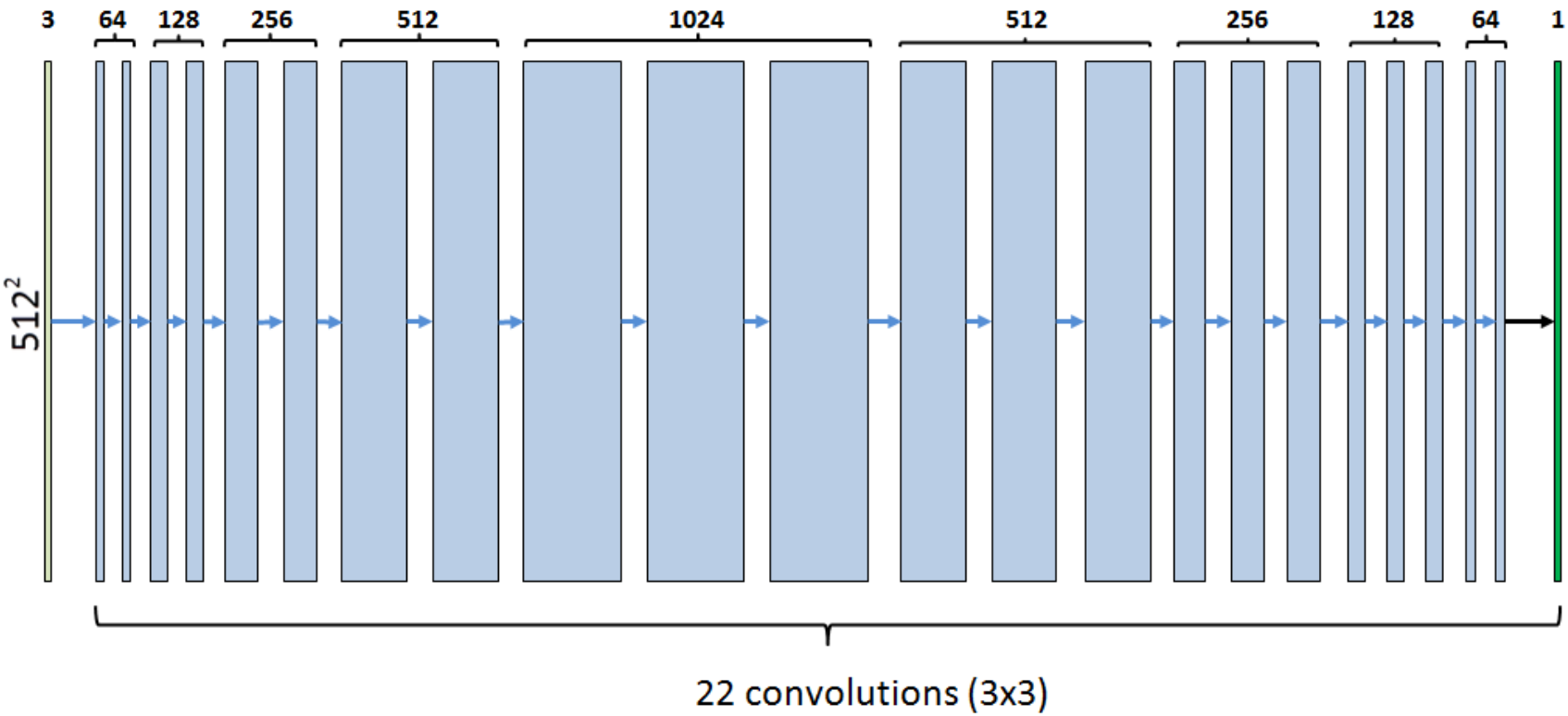


$$Output_{i,j} = \frac{1}{1+exp(-1.7 \cdot \mathbf{w})} \in (0,1)$$

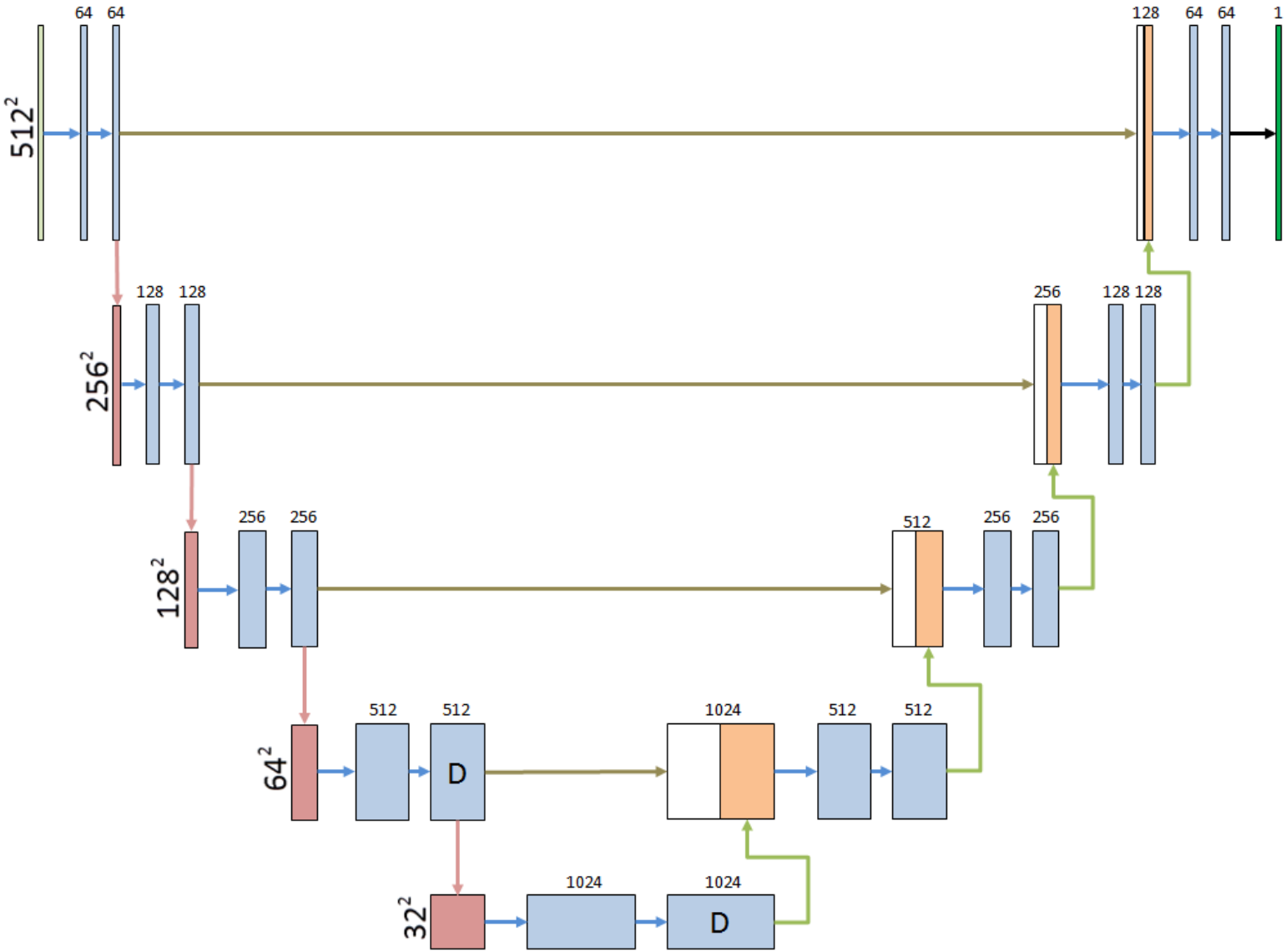
A naive approach..



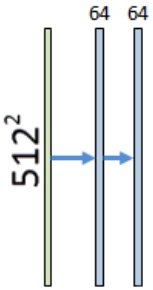
A naive approach..



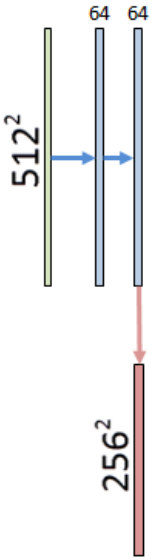
“Our” U-Net


64 64
128 64 64 1
512²
128 128
256 128 128
256²
256 256
512 256 256
128²
512 512
1024 512 512
64²
D
1024 1024
32²
D

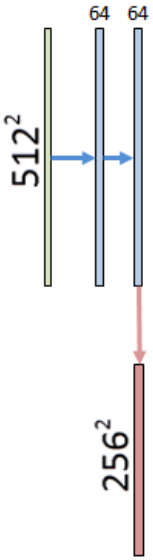
“Our” U-Net



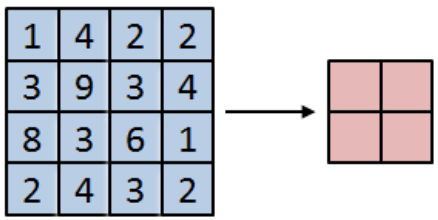
“Our” U-Net



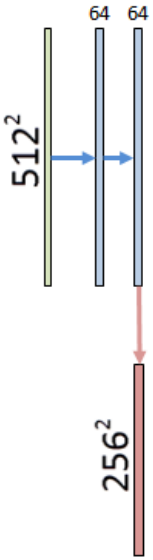
“Our” U-Net



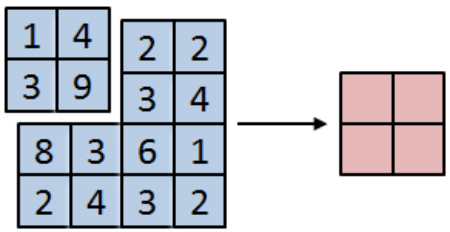
Max pooling:
filter size and stride 2



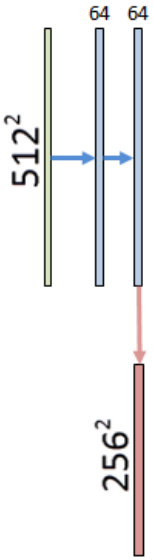
“Our” U-Net



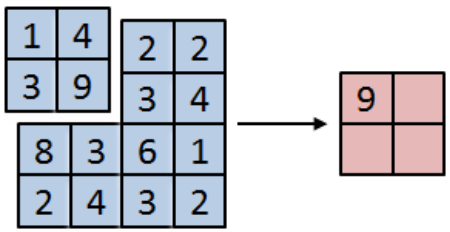
Max pooling:
filter size and stride 2



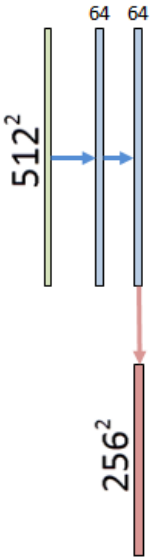
“Our” U-Net



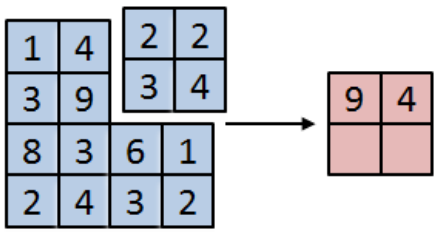
Max pooling:
filter size and stride 2



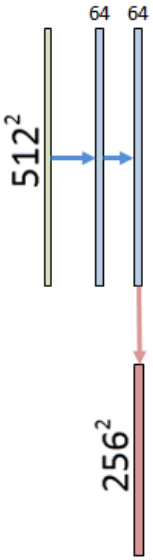
“Our” U-Net



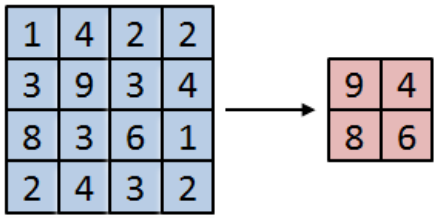
Max pooling:
filter size and stride 2



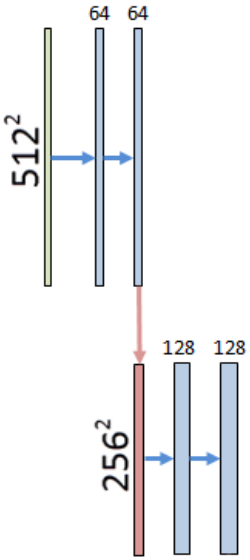
“Our” U-Net



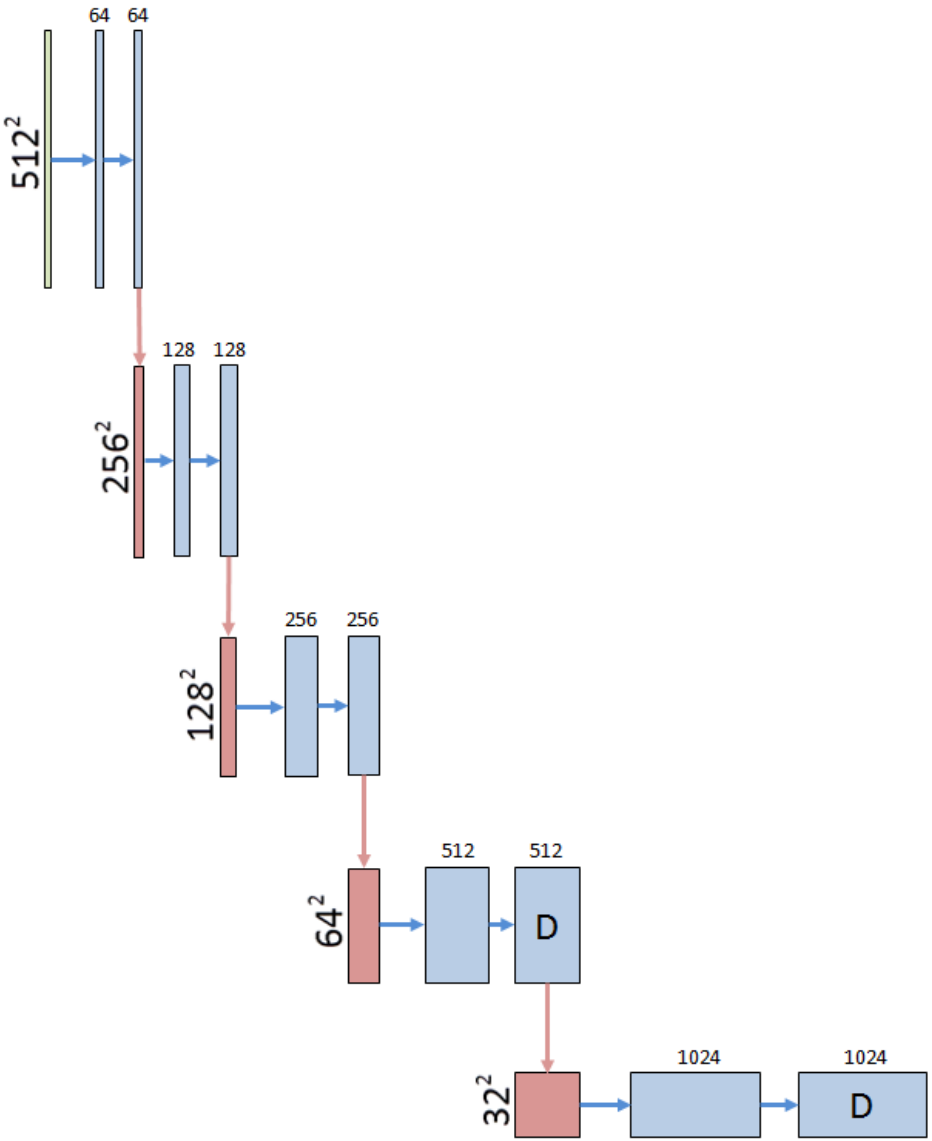
Max pooling:
filter size and stride 2



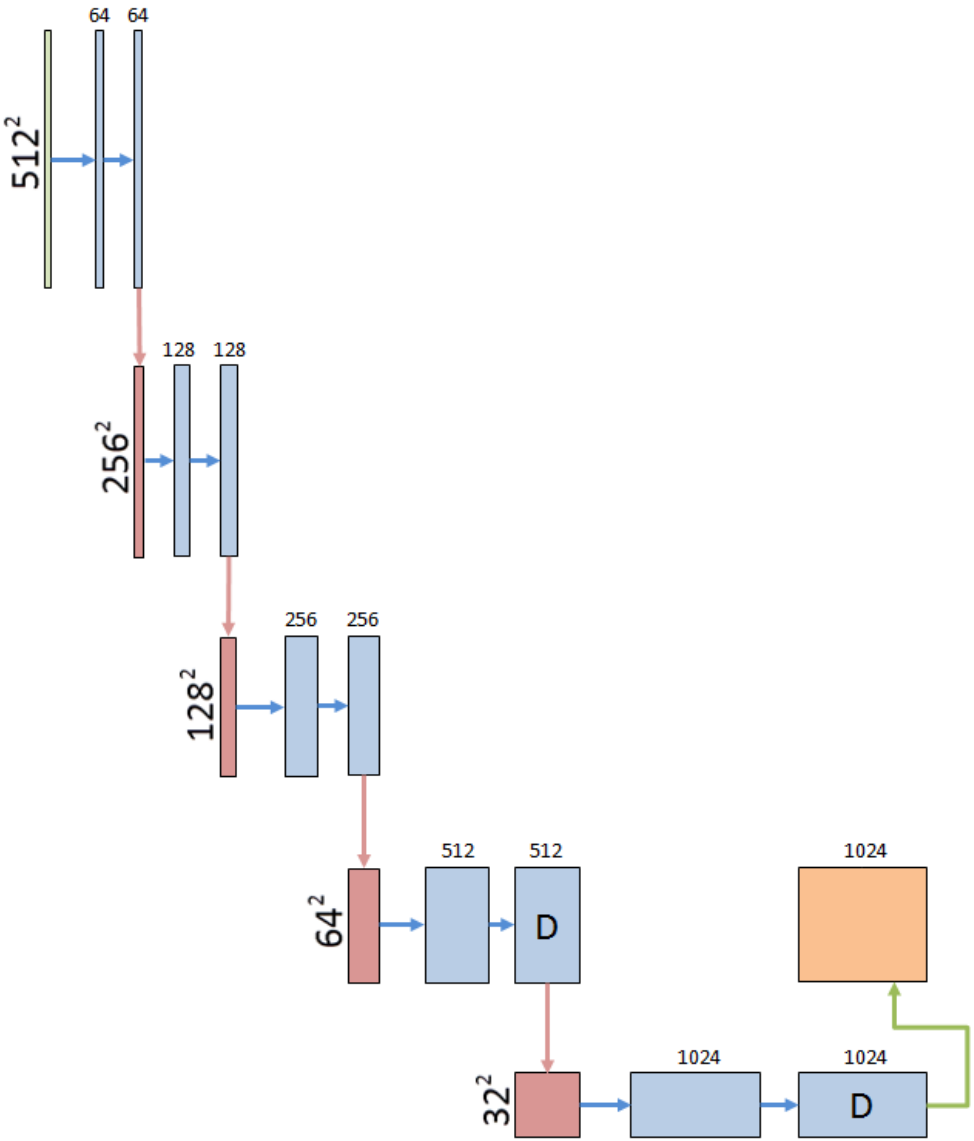
“Our” U-Net



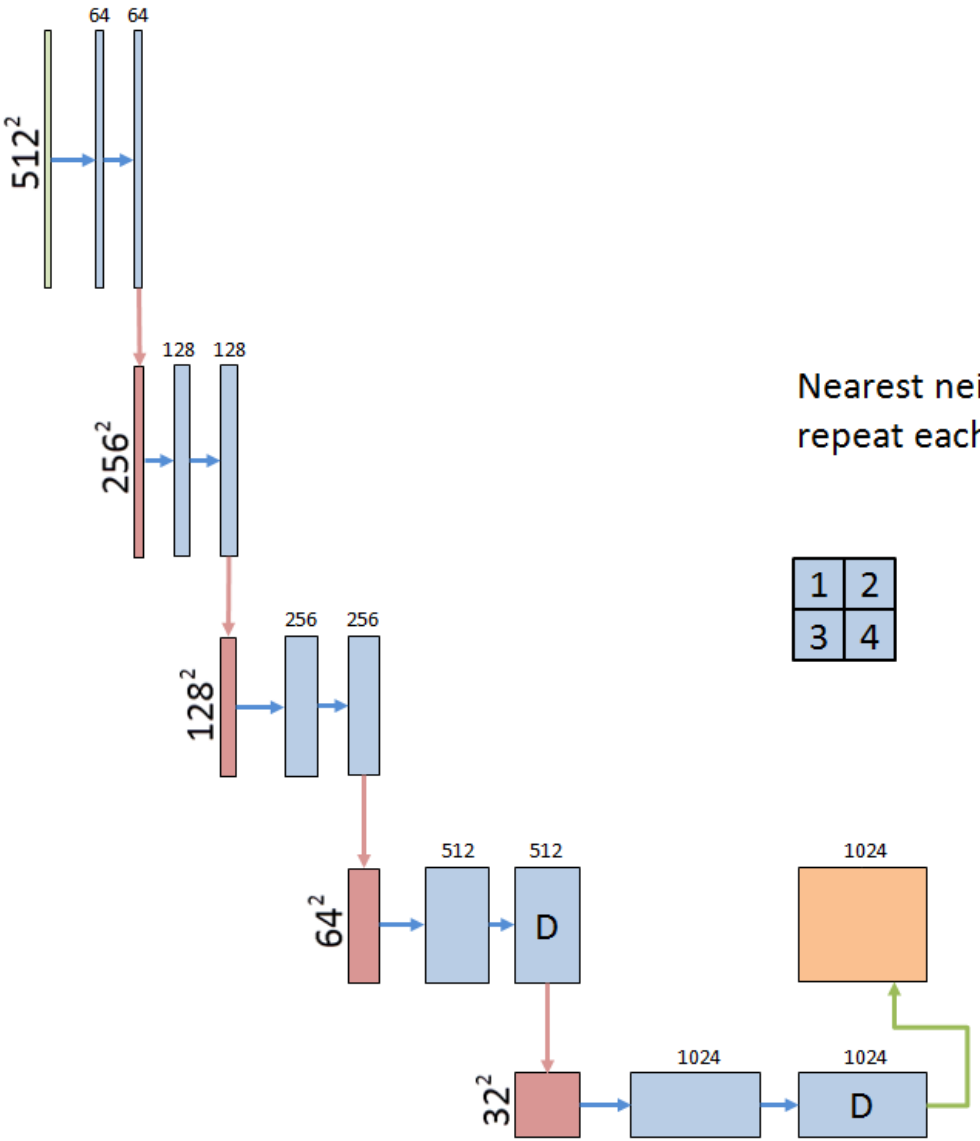
“Our” U-Net


64
64
512^2
128
128
256^2
256
256
128^2
512
512
64^2
D
1024
1024
32^2
D

“Our” U-Net


64
64
512^2
128
128
256^2
256
256
128^2
512
512
D
64^2
1024
1024
1024
D
32^2

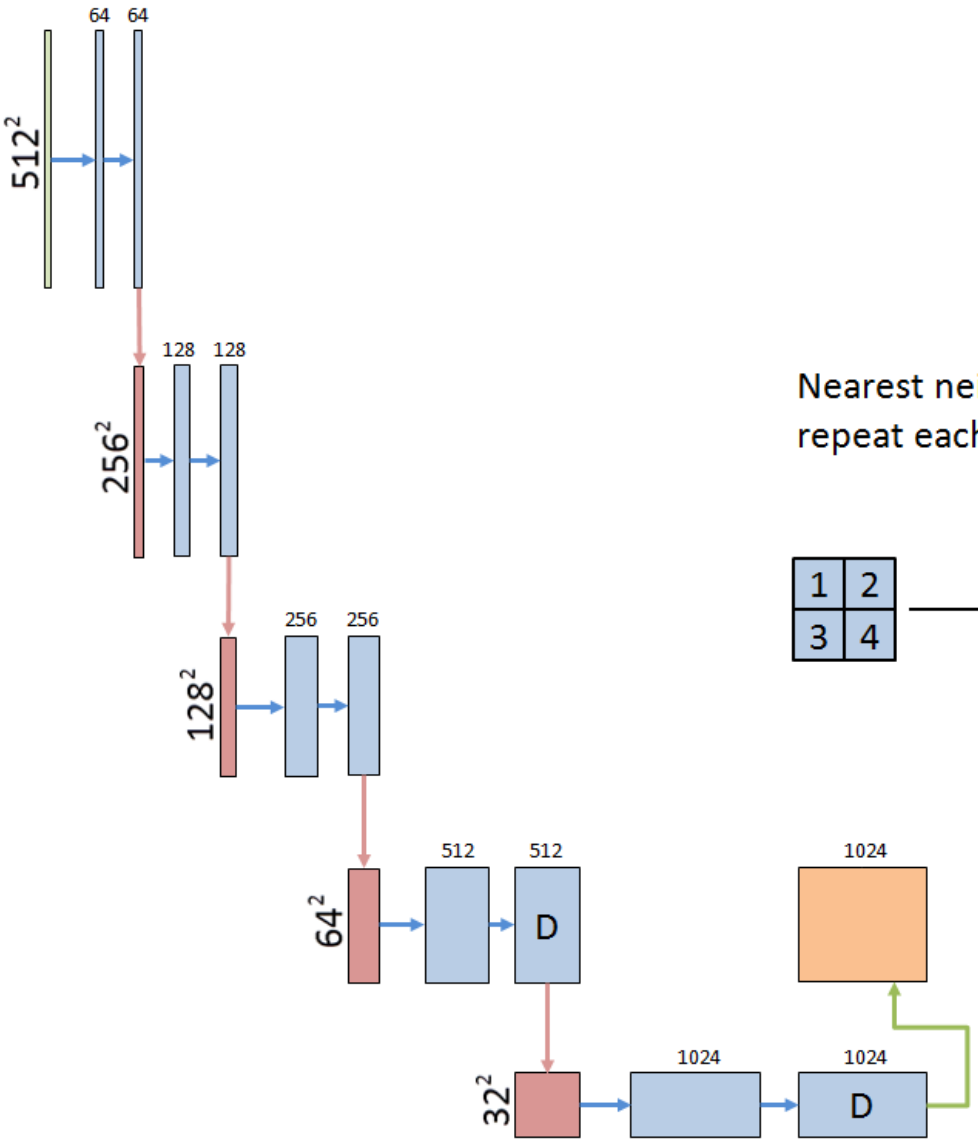
“Our” U-Net



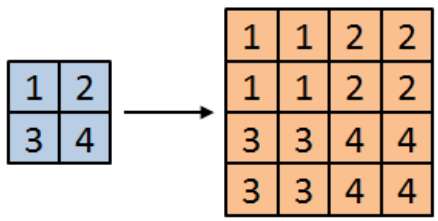
Nearest neighbor interpolation:
repeat each row and column 2 times

1	2
3	4

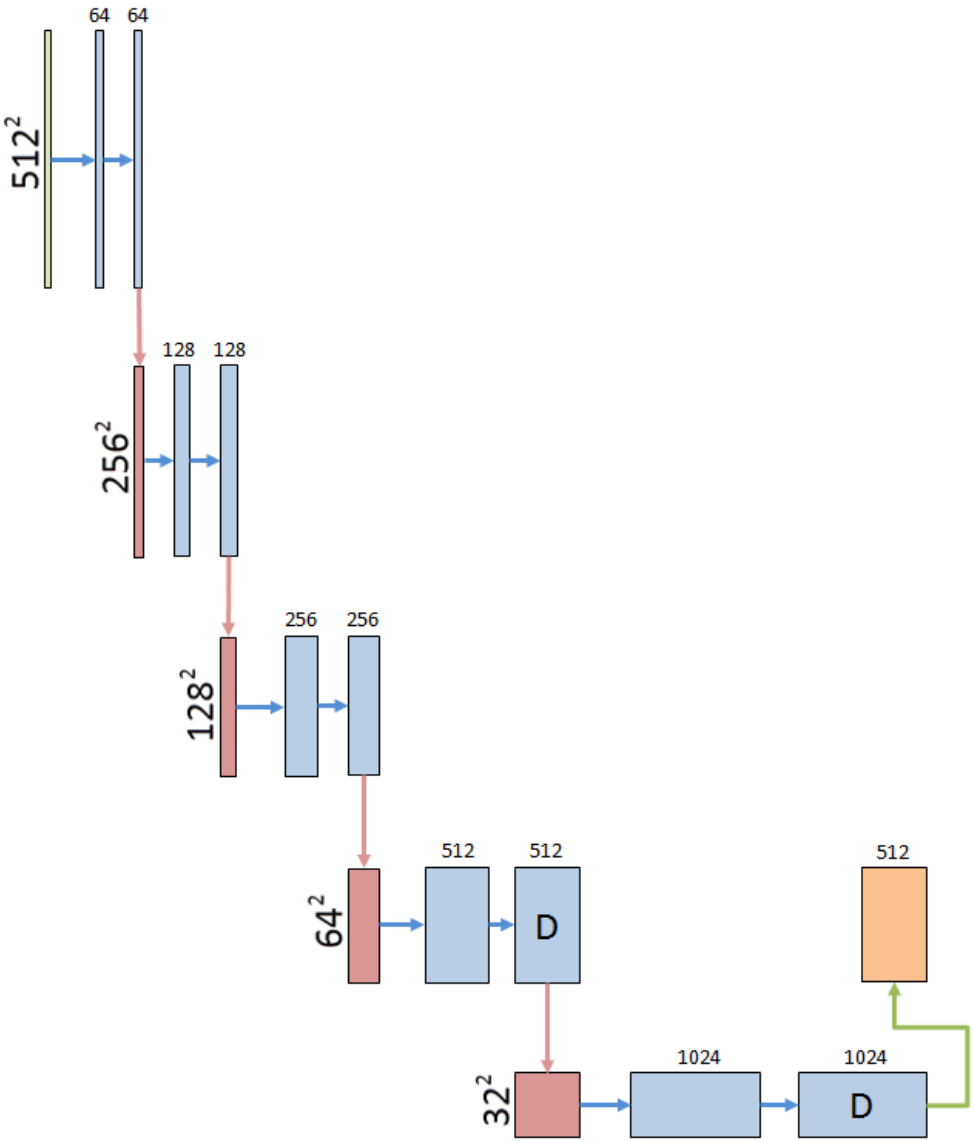
“Our” U-Net



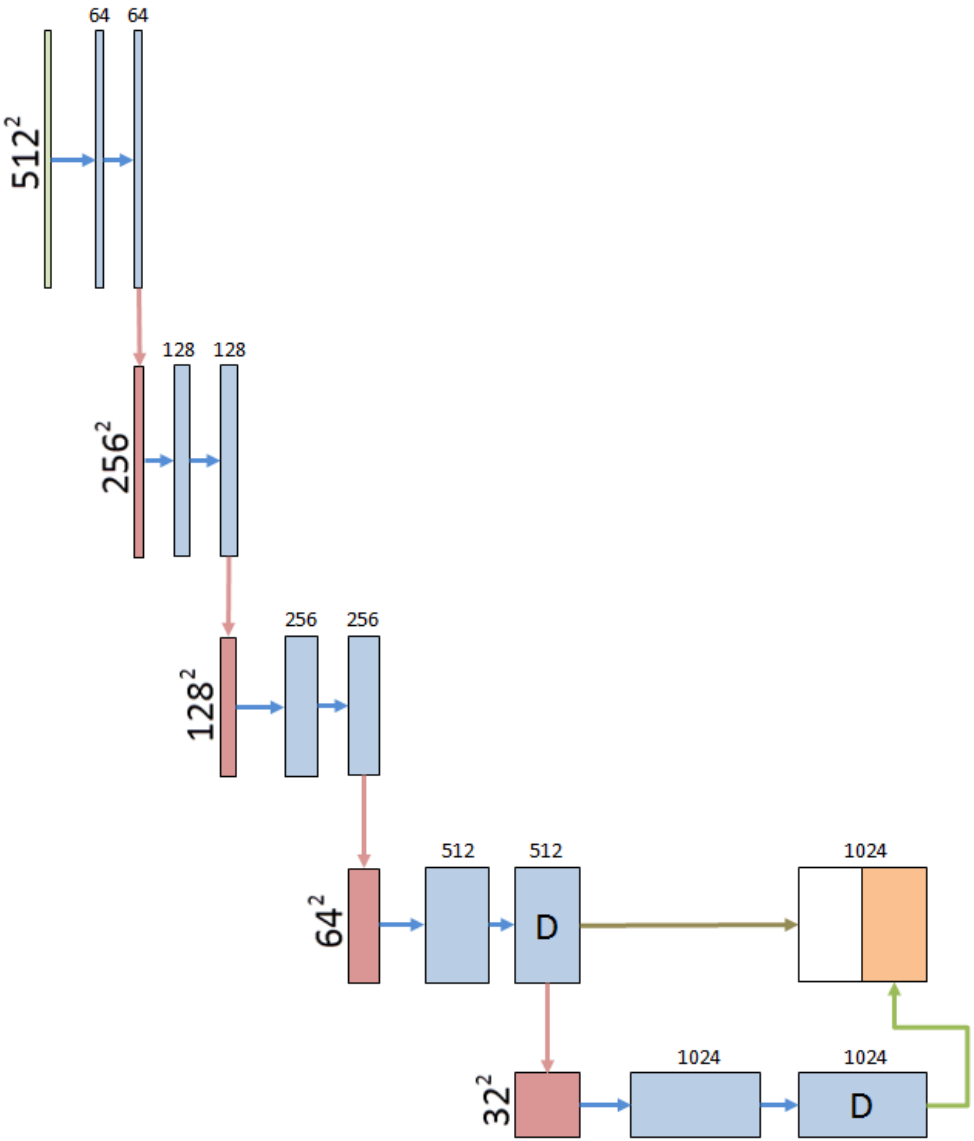
Nearest neighbor interpolation:
repeat each row and column 2 times



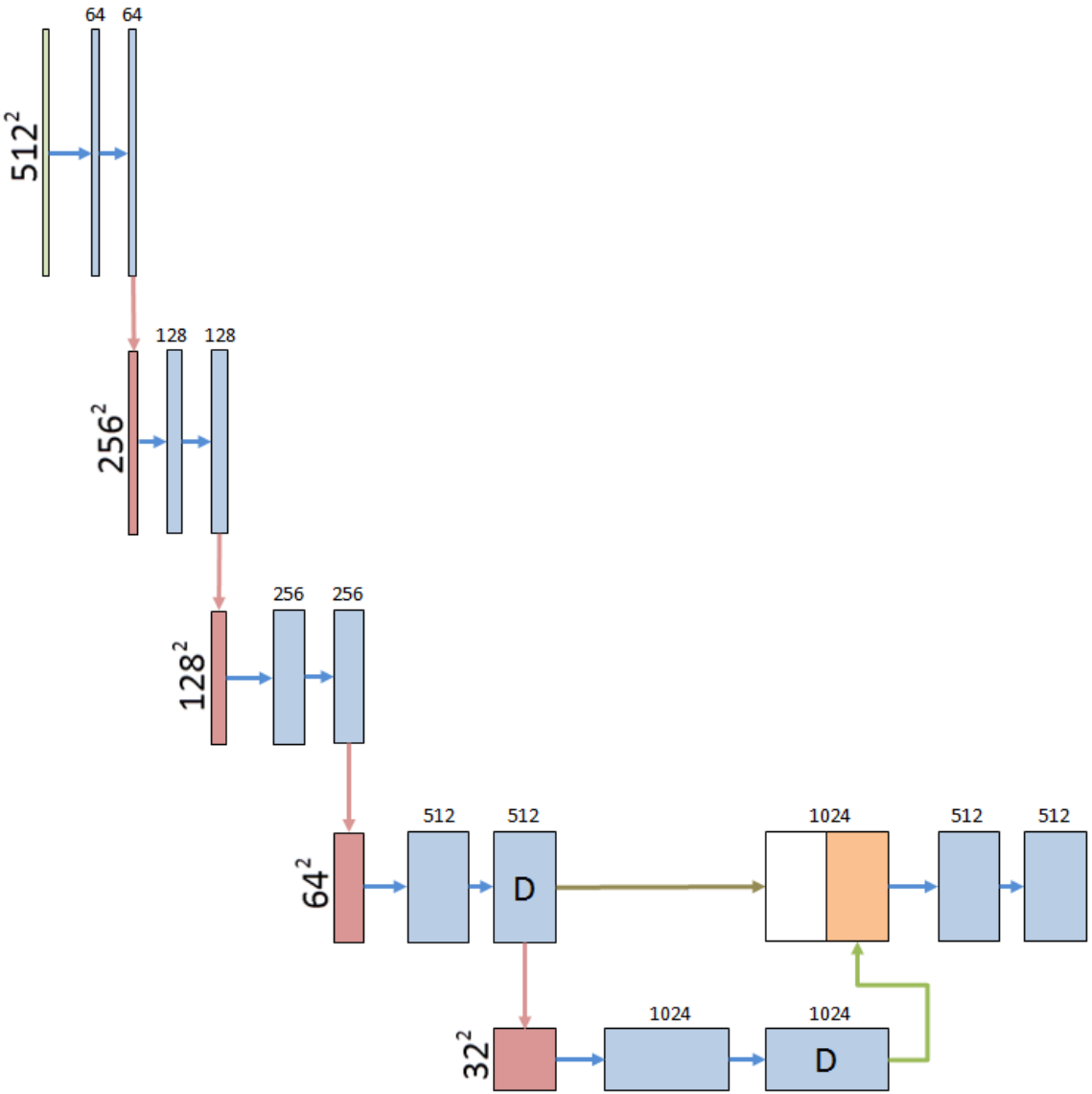
“Our” U-Net


64
64
512^2
128
128
256^2
256
256
128^2
512
512
D
64^2
512
1024
1024
D
32^2

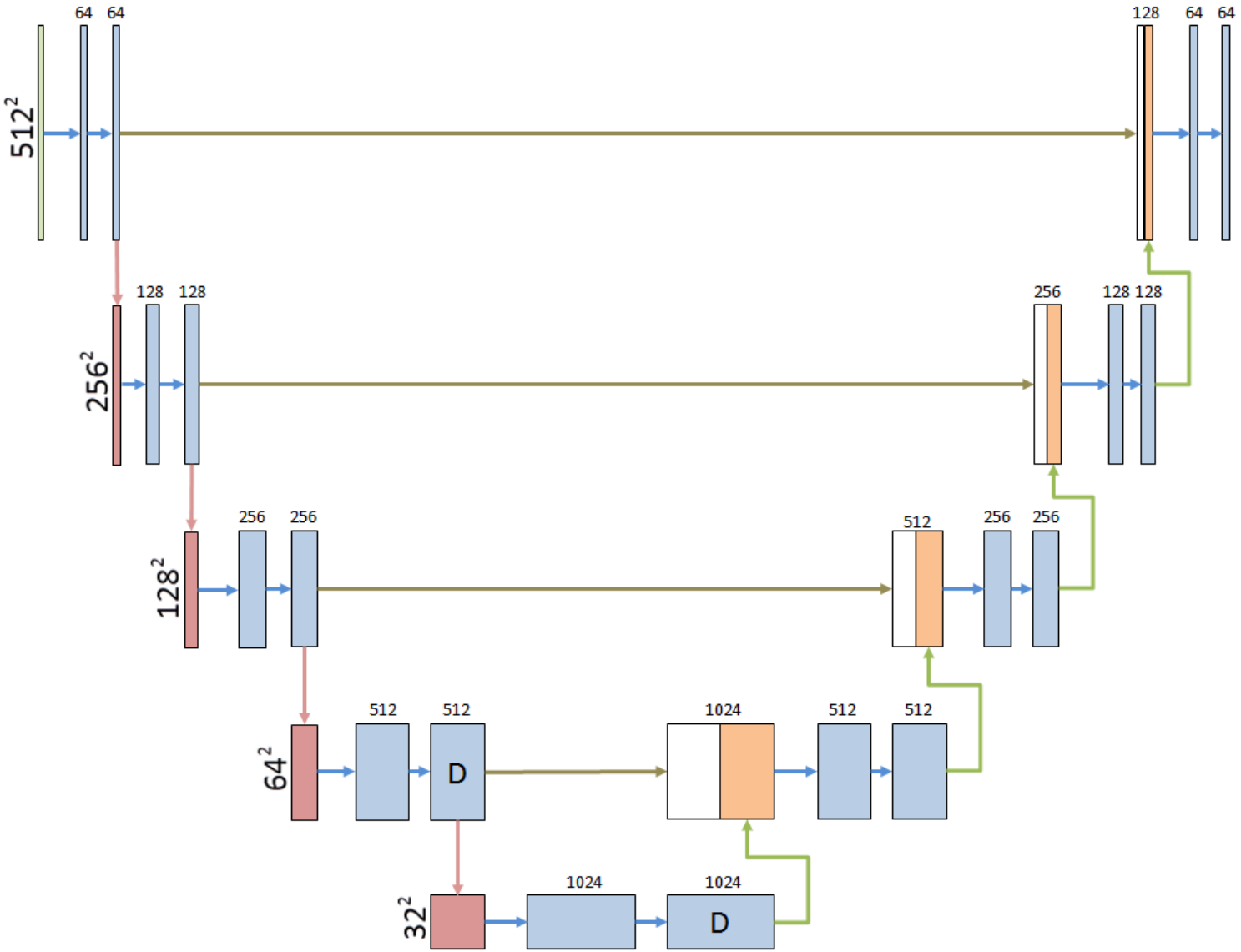
“Our” U-Net


64
64
512^2
128
128
256^2
256
256
128^2
512
512
D
1024
64^2
1024
1024
D
32^2

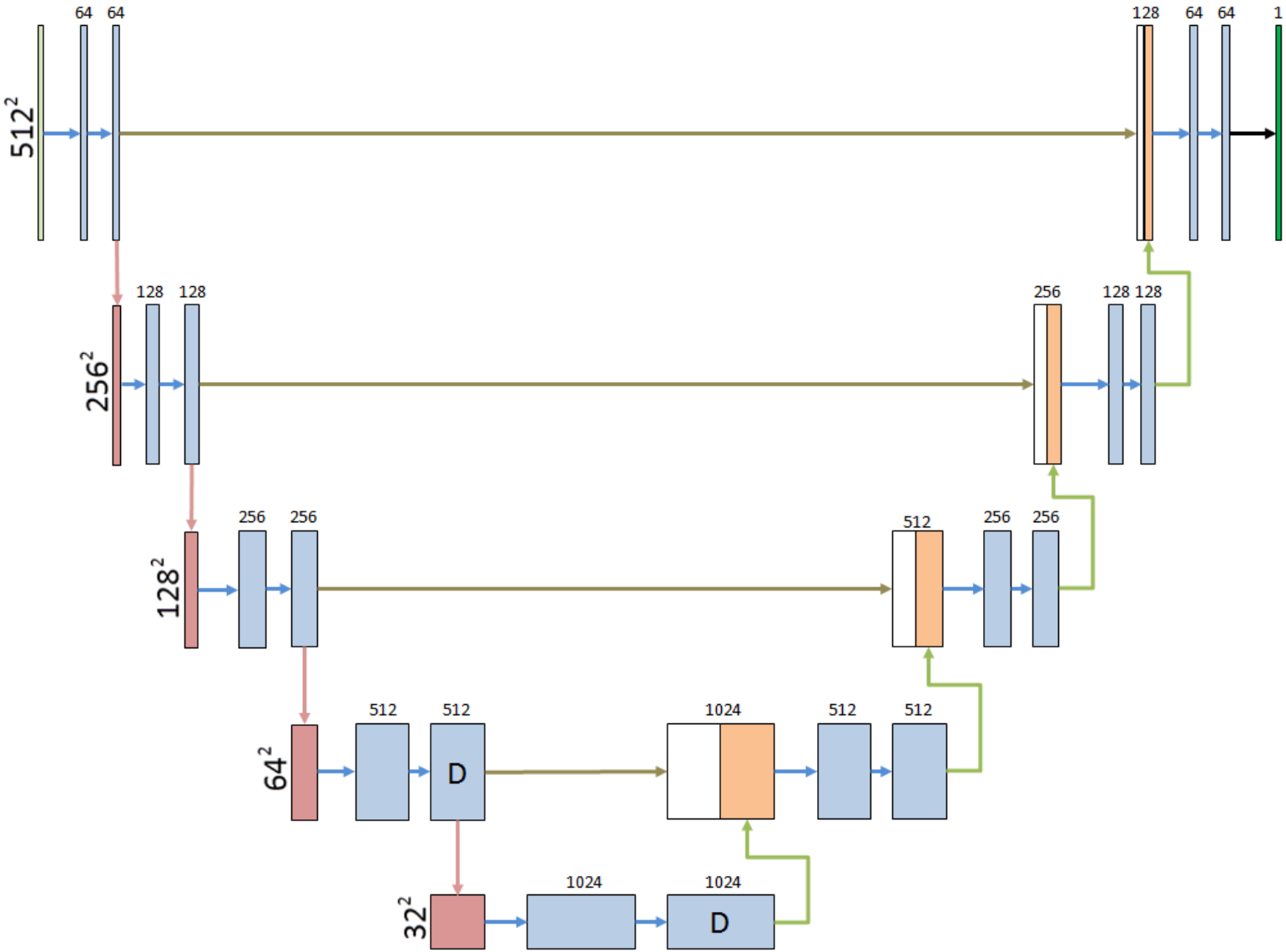
“Our” U-Net


64
64
512^2
128
128
256^2
256
256
128^2
512
512
D
64^2
1024
512
512
1024
1024
D
32^2

“Our” U-Net

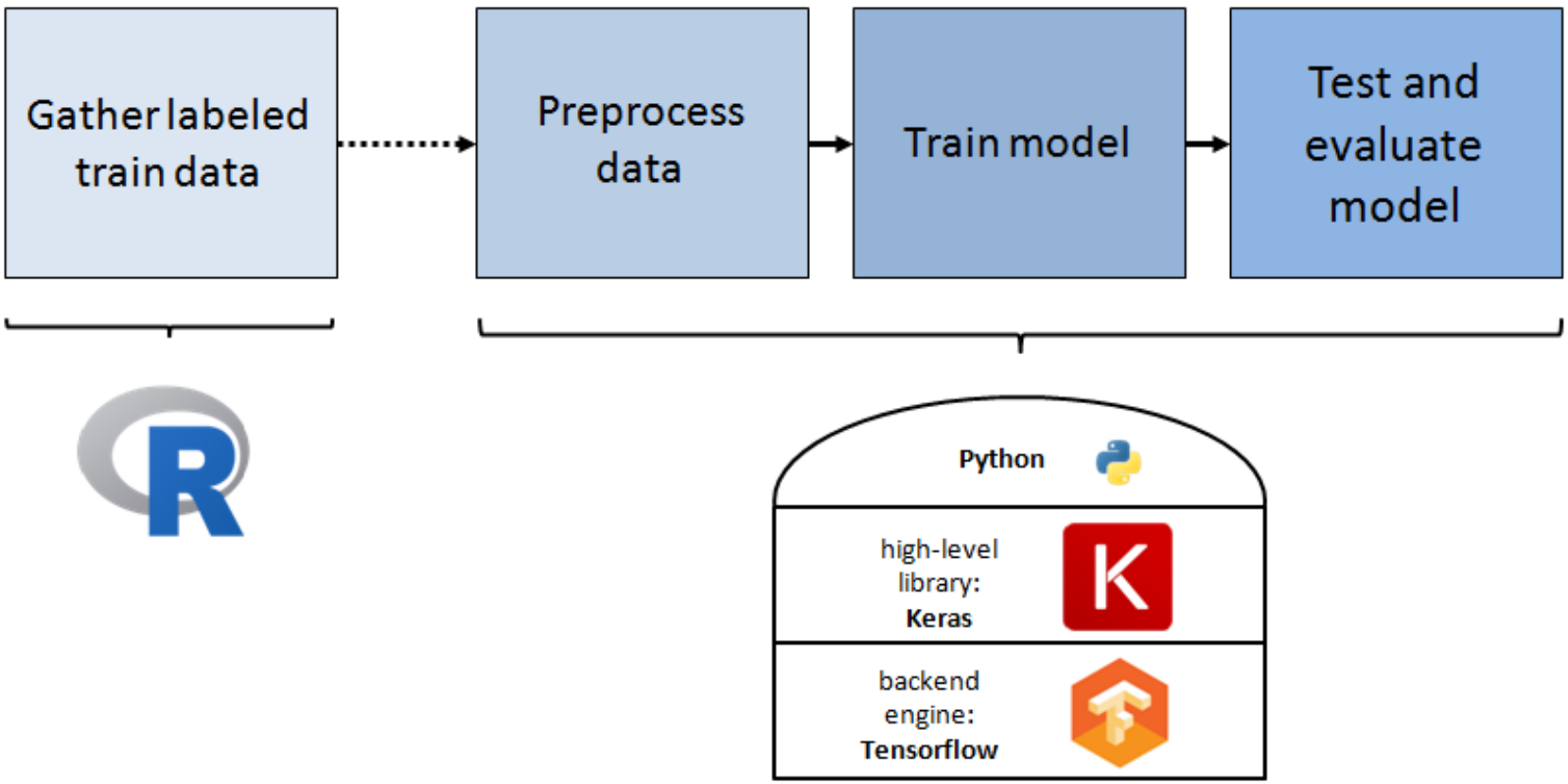

64 64
128 64 64
512²
128 128
256 128 128
256²
256 256
512 256 256
128²
512 512
1024 512 512
64²
D
1024 1024
32²
D

“Our” U-Net


64 64
128 64 64 1
512²
128 128
256 128 128
256²
256 256
512 256 256
128²
512 512
1024 512 512
64²
D
1024 1024
32²
D

Structure

Pipeline


Gather labeled train data
Preprocess data
Train model
Test and evaluate model
Python
high-level library: Keras
backend engine: Tensorflow

Training and hyperparameters

- Due to hardware and temporal limitations, we had to select the train data very carefully.
- We used a thresholder, such that images as well as their labels with less than 5% pixels amounting to roads were sorted out.

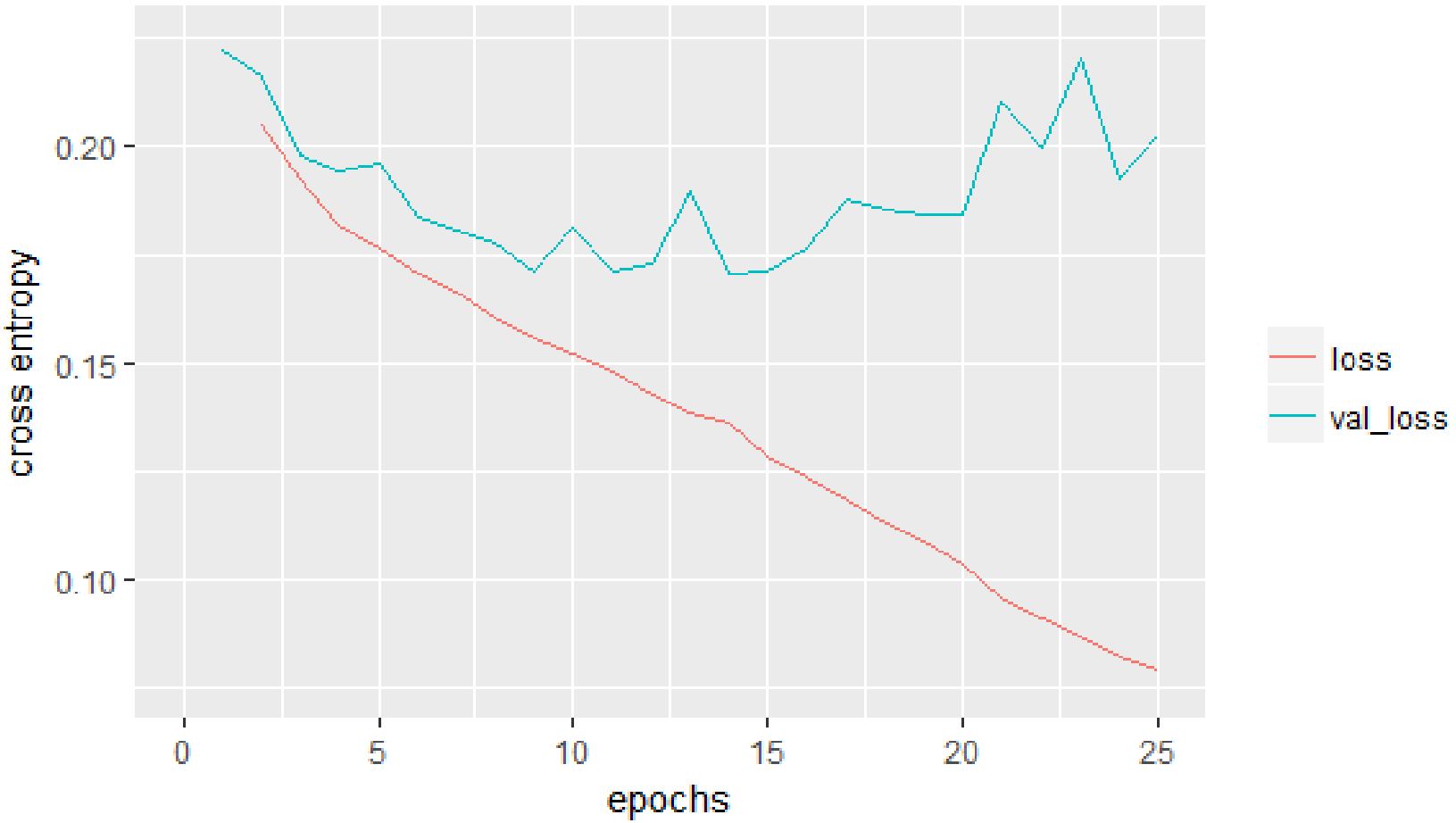
Training and hyperparameters

- Of roughly 5000 Images, 873 “survived” the thresholder

Training and hyperparameters

- Of roughly 5000 Images, 873 “survived” the thresholder

Training and validation



$n_{\text{train}} = 720$, $n_{\text{val}} = 80$, GPU: 300 s/epoch, CPU: 8000 s/epoch

Error measure

	Actual positive	Actual negative	
Predicted positive	True positive (TP)	False positive (FP, Type I error)	Precision (P) $= \frac{\#TP}{\#TP + \#FP}$
Predicted negative	False negative (FN, Type II error)	True negative (TN)	
	Recall (R) $= \frac{\#TP}{\#TP + \#FN}$		Accuracy $= \frac{\#TP + \#TN}{n}$

- F_1 score as the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

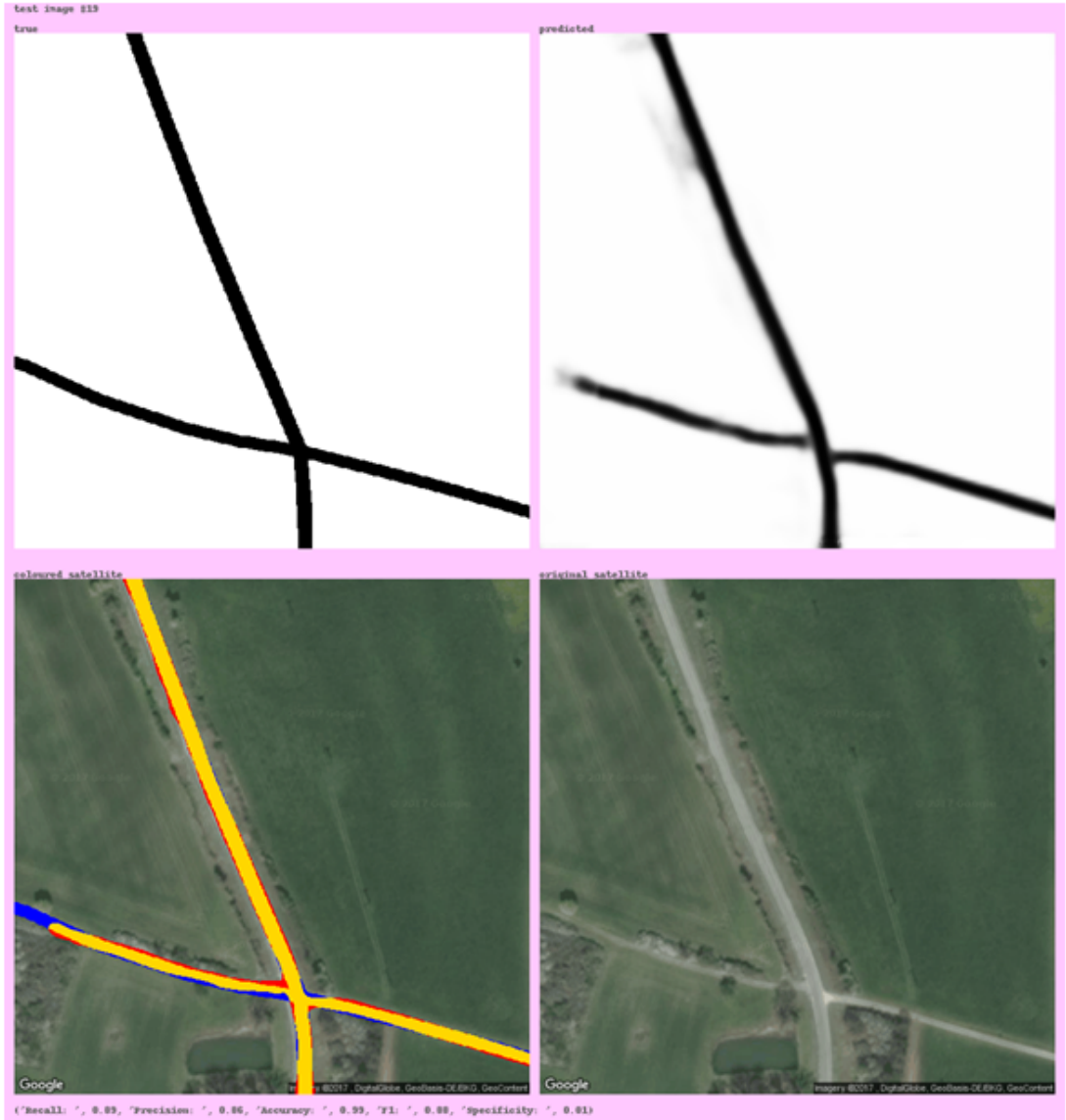
Test results I

- Test results on 40 images:

Recall	Precision	F1 Score	Accuracy	Base Accuracy
0.7830	0.7177	0.7439	0.9559	0.9208

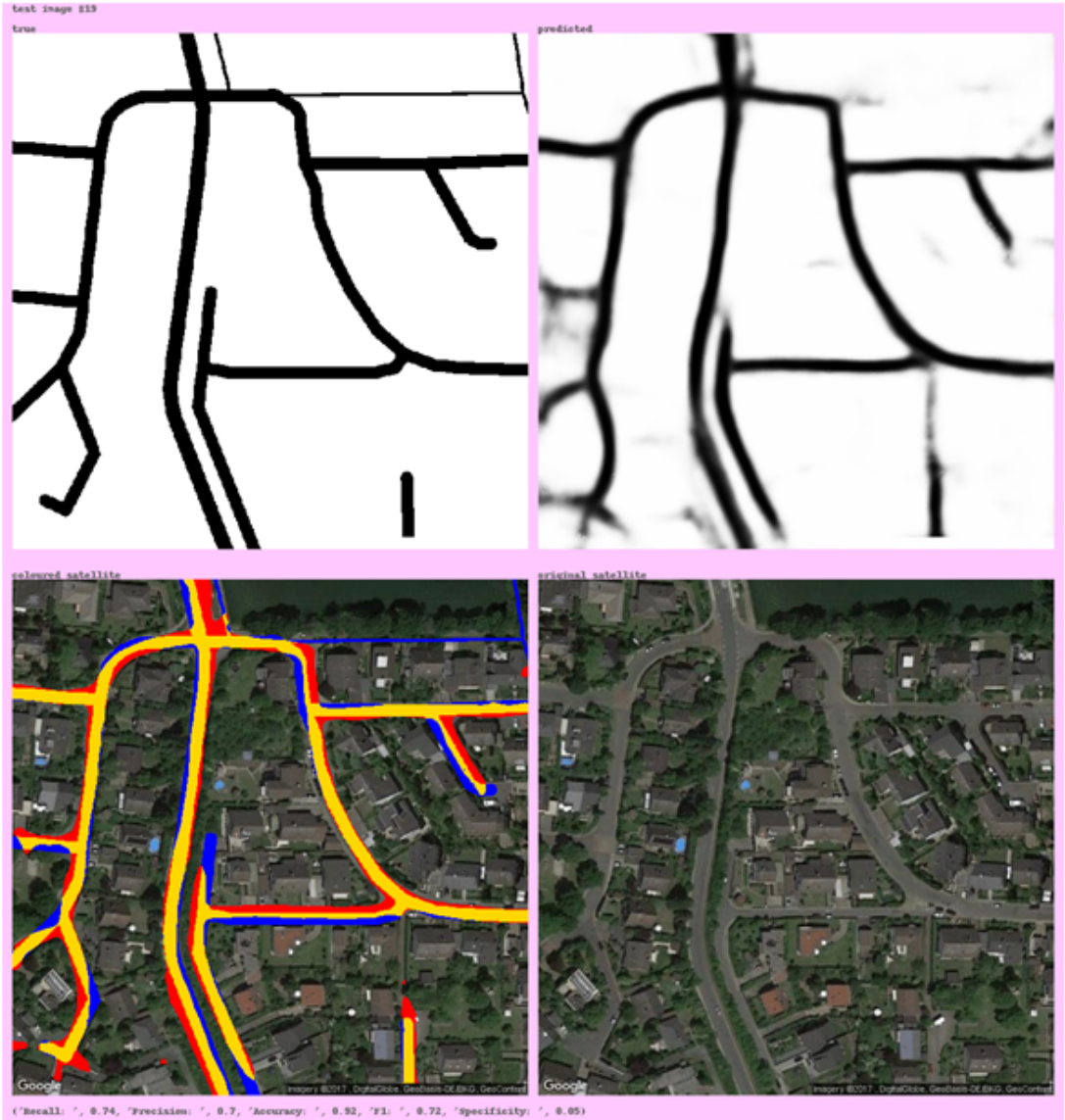
- Interpretation:
  - Recall: 78.30% of the labeled road pixels in the test data set were classified as such by the model.
  - Precision: 71.77% of those pixels, that were classified as road pixels, are also labeled as such.

Test results II



Recall: 0.89
Precision: 0.86
F1: 0.88

Test results II



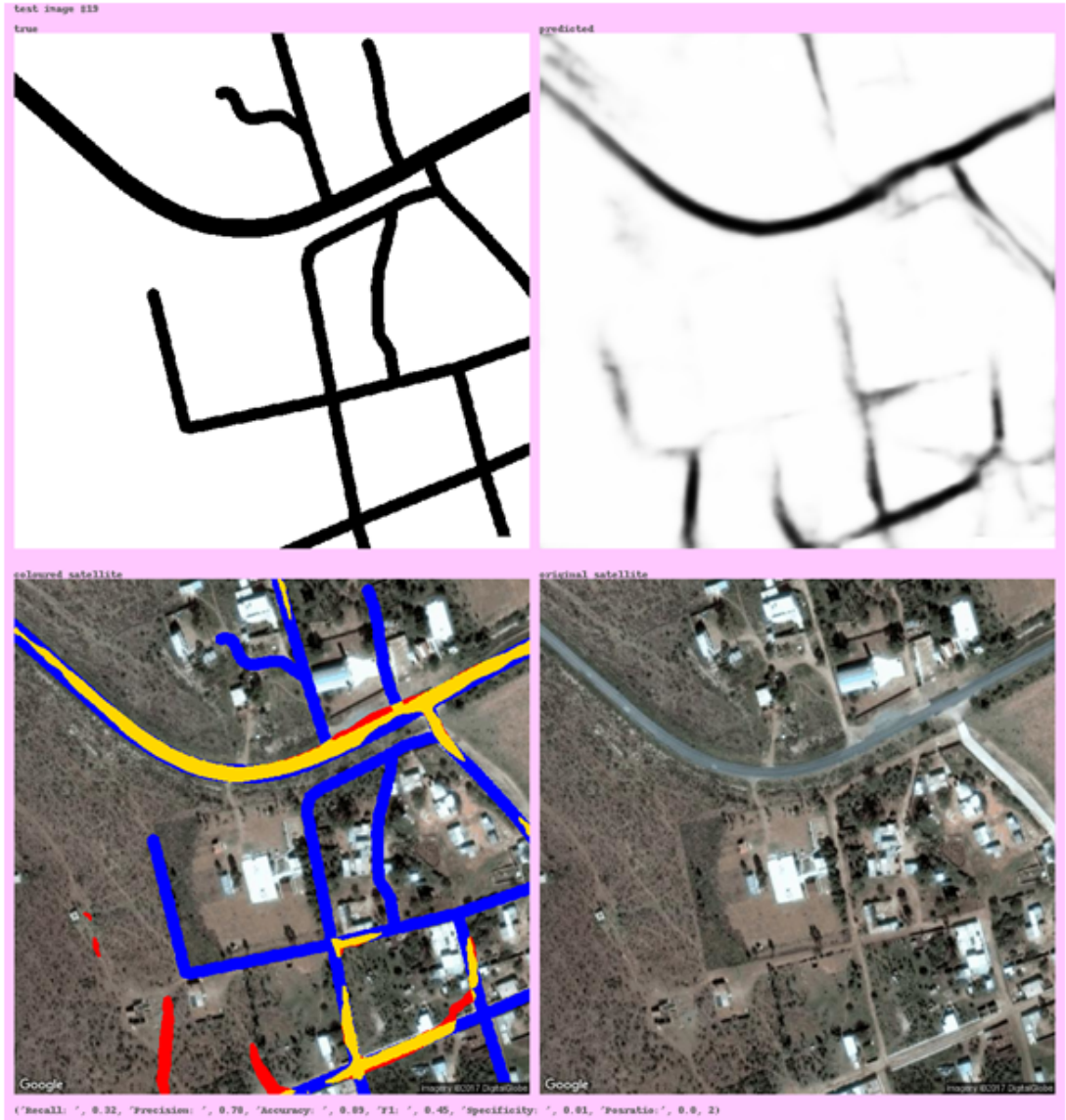
Recall: 0.74
Precision: 0.7
F1: 0.72

Q1: Influence of environmental landscape I

- Test our model trained on german satellite imagery on mexican images
- Test result on 20 images:

Country	Recall	Precision	F1 Score	Accuracy	Base Accuracy
Mexico	0.3802	0.5130	0.4149	0.9393	0.9360
Germany	0.7830	0.7177	0.7439	0.9559	0.9208

Q1: Influence of environmental landscape II



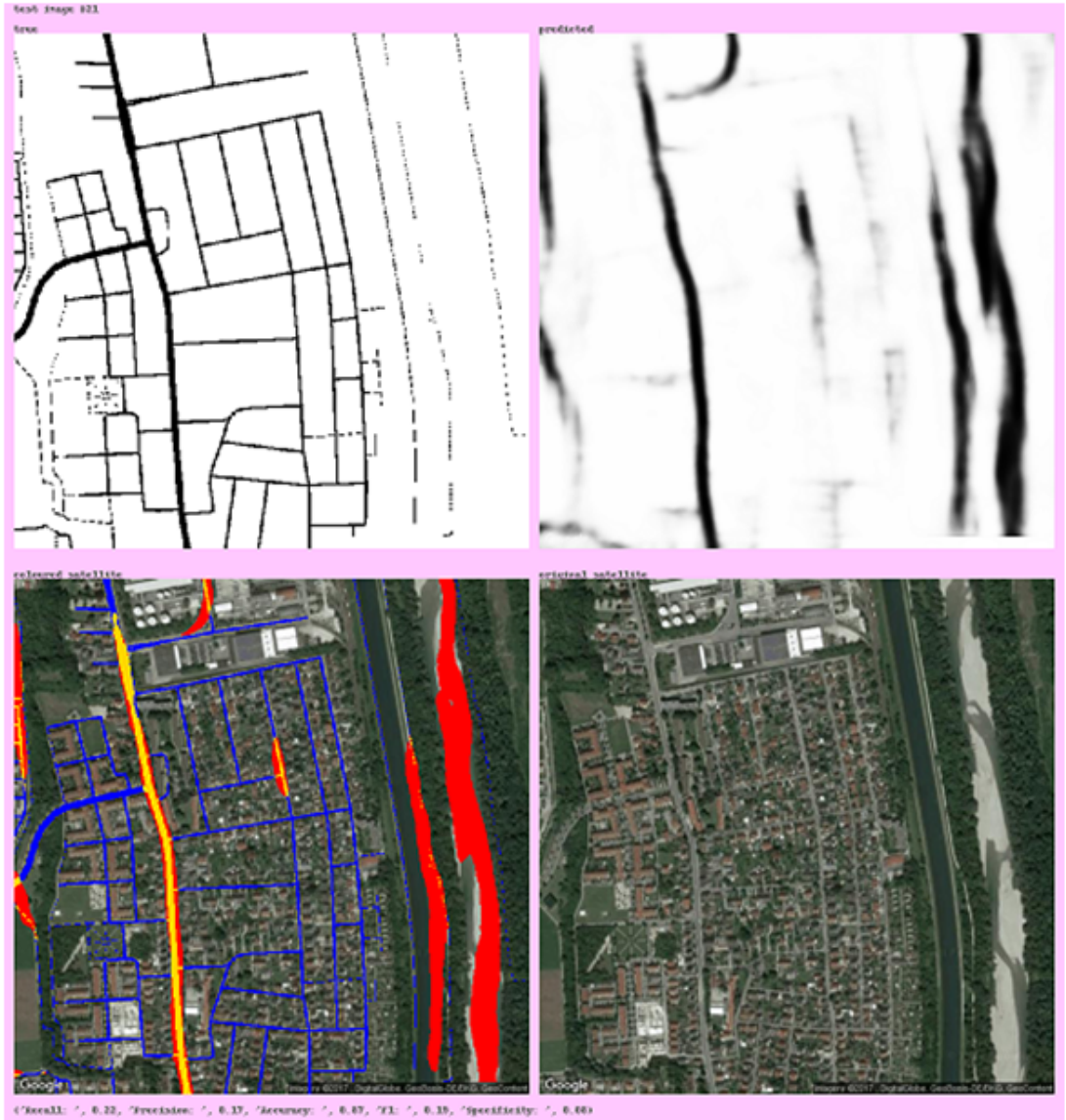
Recall: 0.32
Precision: 0.78
F1: 0.45

Q2: Influence of zoom I

- Suppose we would apply our model on data with further zoom level
- Test results on 100 images:

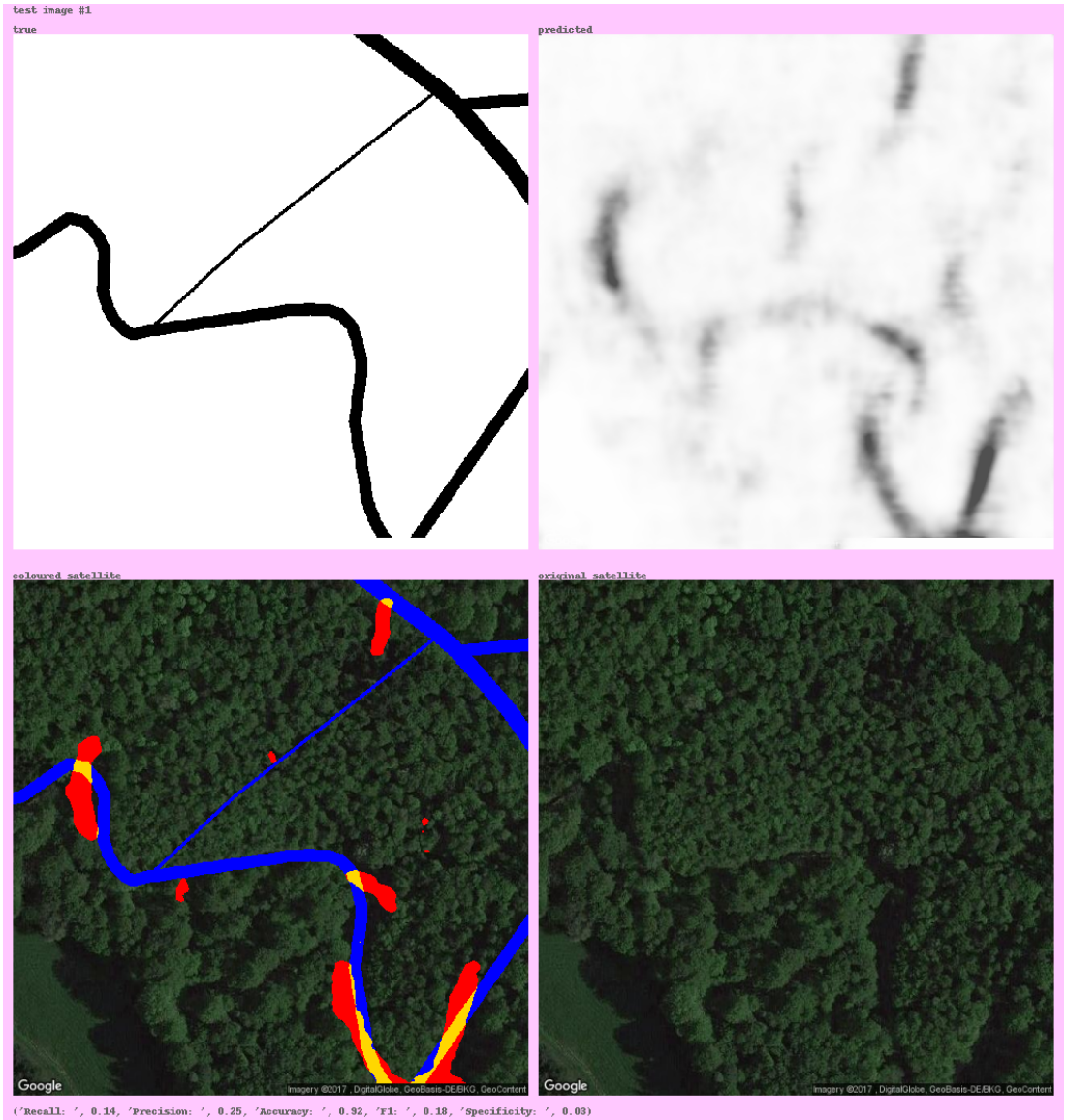
Test data	Recall	Precision	F1 Score	Accuracy	Base Accuracy
Zoom 16	0.3931	0.2595	0.2869	0.9568	0.9729
Zoom 18	0.7830	0.7177	0.7439	0.9559	0.9208

Q2: Influence of zoom II

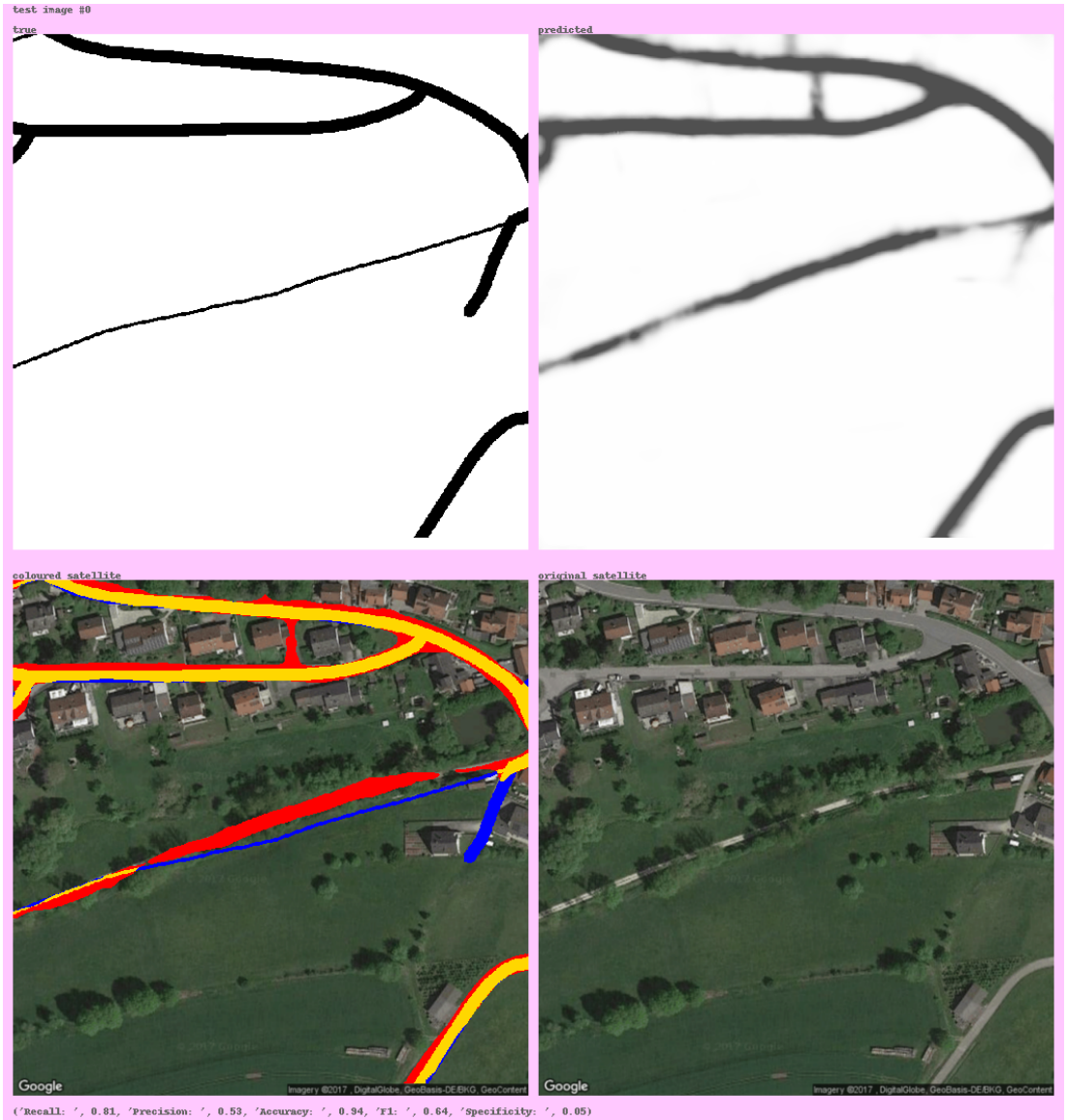


Recall: 0.22
Precision: 0.17
F1: 0.19

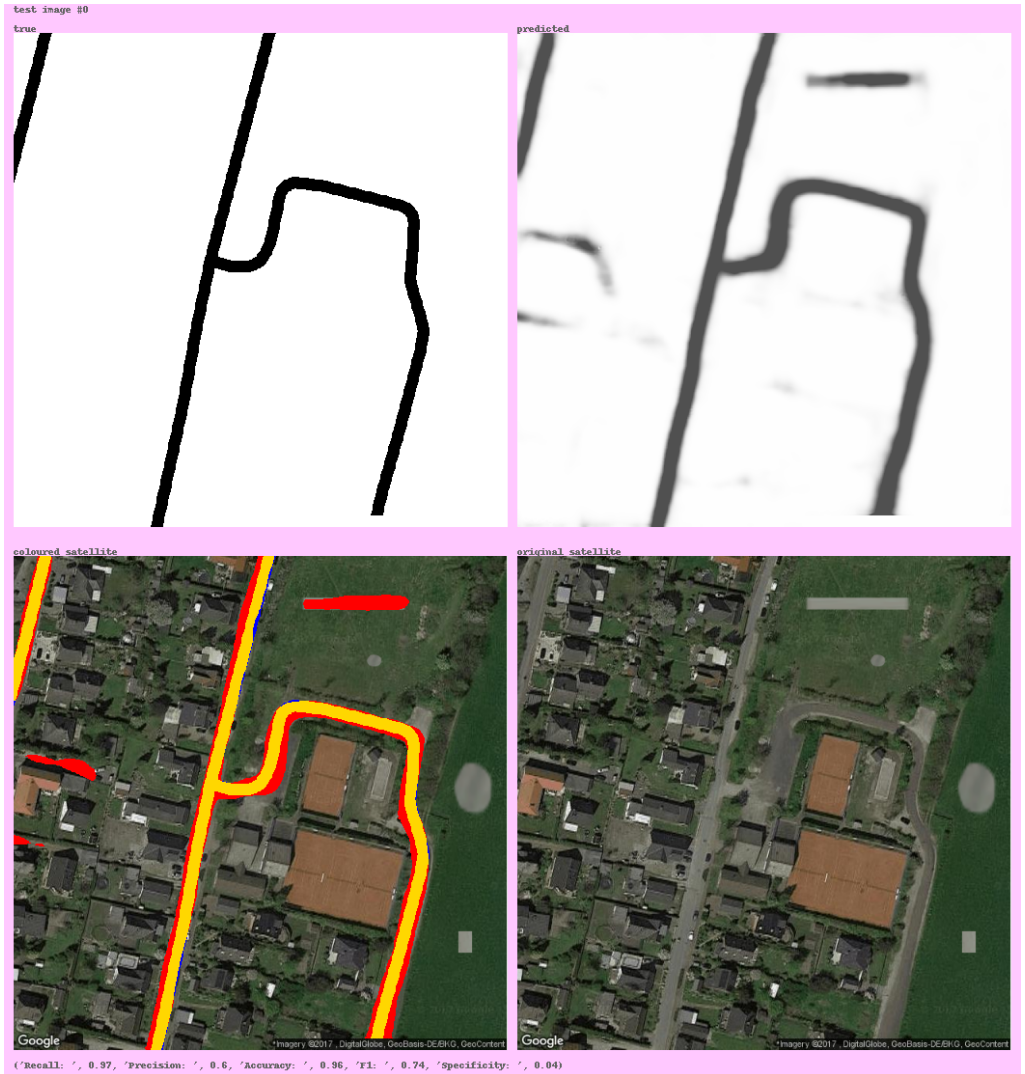
Q3: Importance of train data quality



Q3: Importance of train data quality



Q4: Stability of the model



Q5: Urban vs. rural areas I

- Test on urban and rural areas
- 2 datasets with 20 images each
- Resulting average values:

Model	Recall	Precision	Accuracy	F1 Score	Base Accuracy
Urban	0.7784	0.6804	0.9472	0.7221	0.9208
Rural	0.7876	0.7550	0.9647	0.7658	0.9276

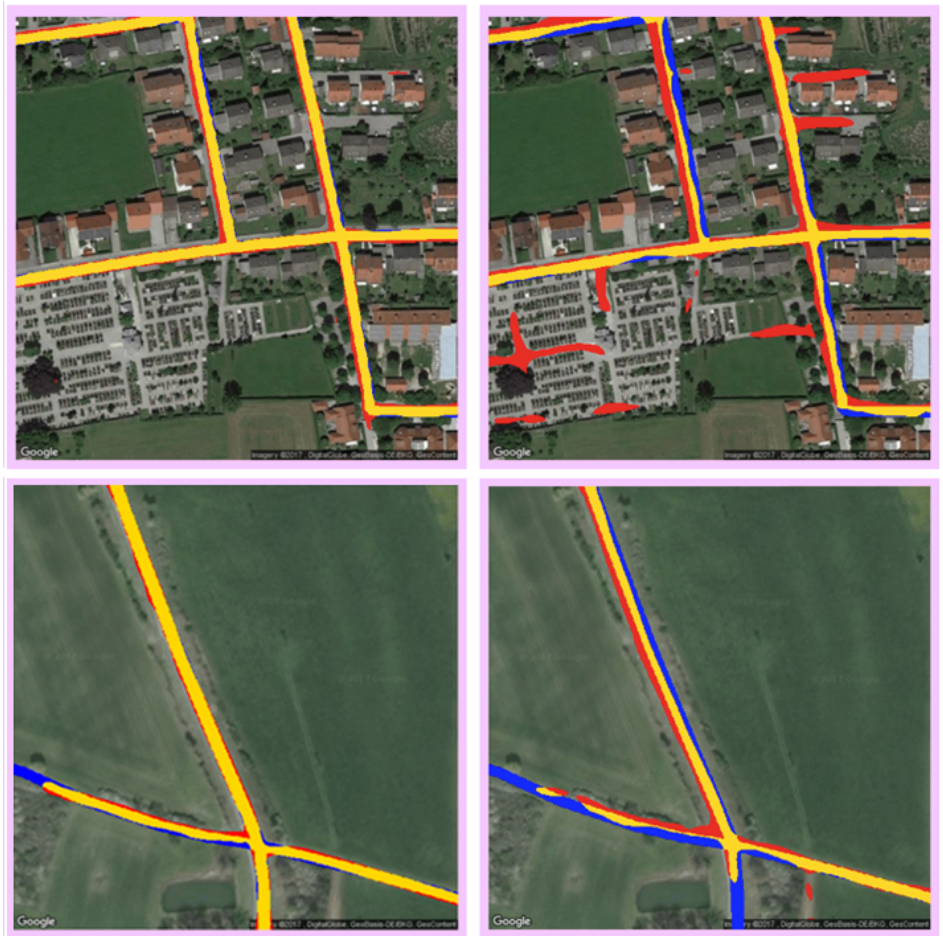
Q6: Amount of train data

- Train on small (200) and big (800) dataset
- Test on same dataset
- Results:

Model	Recall	Precision	Accuracy	F1 Score	Early Stopping
Small	0.6782	0.7031	0.9506	0.6811	38
Big	0.7830	0.7177	0.9559	0.7439	25

Q6: Amount of train data

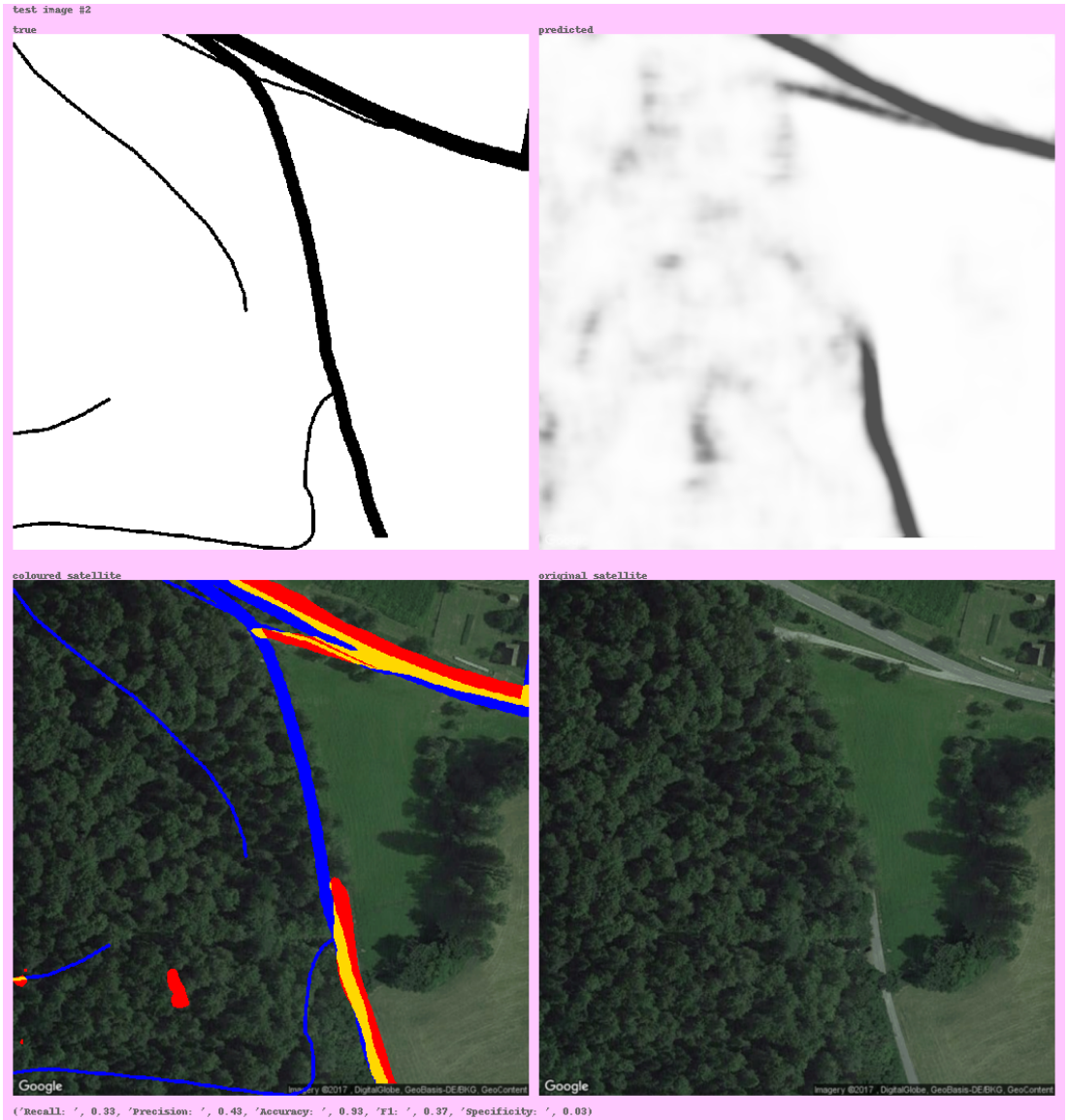
- Net gets more confident with increasing amount of train data



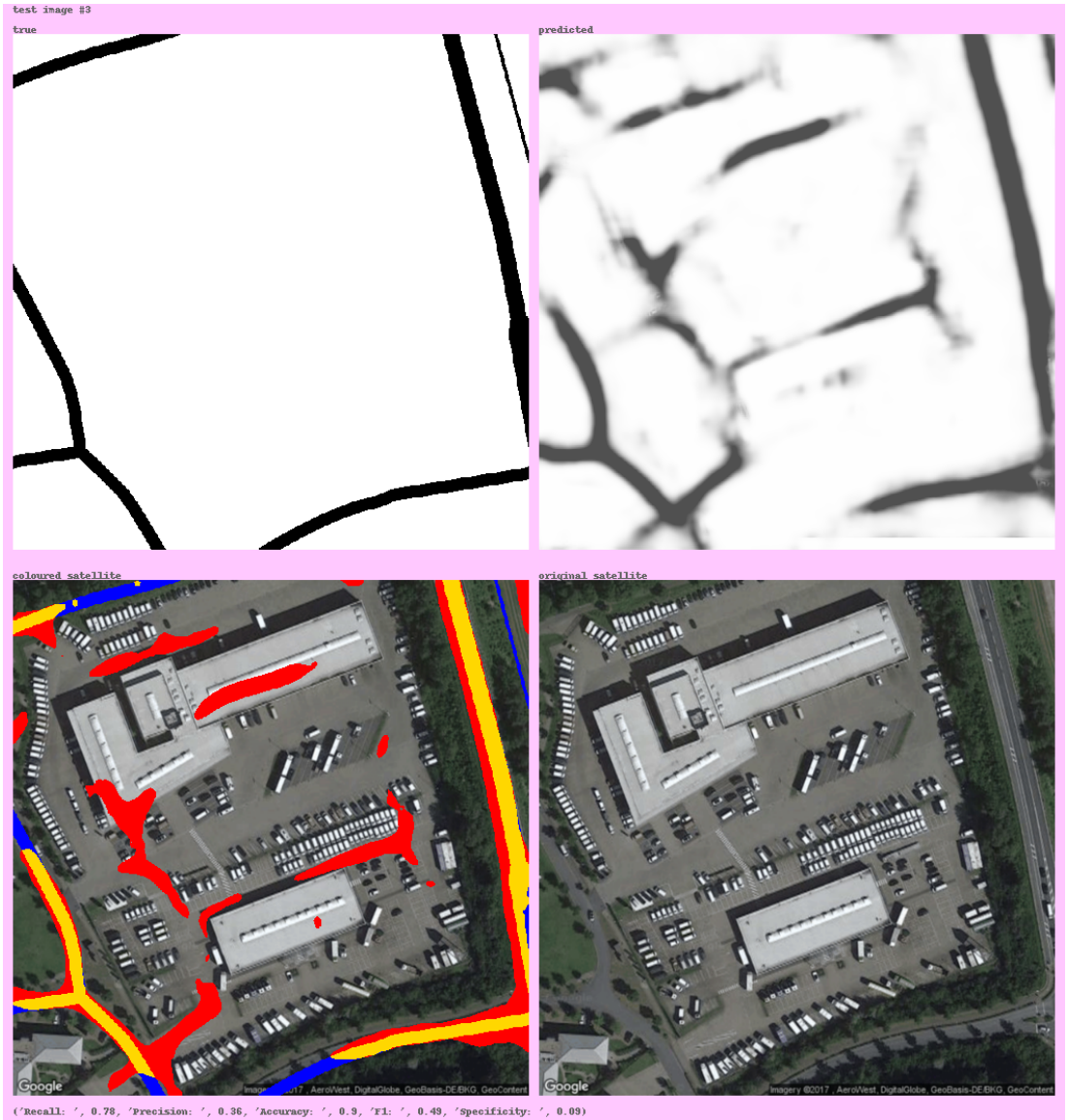
Left: prediction with big net, right: prediction with small net

Q7: Difficulties

Q7: Difficulties



Q7: Difficulties



Structure

Outlook and extensions

- Surprisingly often, our network performs better than the actual label pretends
- Try tailored loss functions (to overcome unbalanced class problem)
- Gather multi-labeled train data to solve difficulties (e.g. railroads)
- Use dataset augmentation (use high performance hardware to train on massive data)
- Find additional methods to clean train data automatically

Our Code..

- is available at:
  `https://github.com/Goschjann/road_segmentation_project`
- the repo includes
  - a script to download a public data set
  - a script for training u-net
  - and a script to test and visualize the results

Thank you!

References I

Mnih Volodymyr (2013)

Machine Learning for Aerial Image Labeling

https://www.cs.toronto.edu/~vmnih/docs/Mnih_Volodymyr_PhD_Thesis.pdf

Fei-Fei Li, Justin Johnson, Serena Yeung (2017)

Detection and Segmentation

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf (visited 22.05.2017)

Joel Janai, Fatma Güne, Aseem Behl, Andreas Geiger (2017)

Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art

http://arxiv.org/abs/1704.05519

Olaf Ronneberger, Philipp Fischer, Thomas Brox (2015)

U-Net: Convolutional Networks for Biomedical Image Segmentation

http://arxiv.org/abs/1505.04597

References II

Michael H. Herzog, Aaron M. Clarke (2014)

Why vision is not both hierarchical and feedforward

https://www.scienceopen.com/document?id=2d184a64-d52e-42b4-9acf-25ba1e637a38

Yann LeCun, Marc'Aurelio Ranzato (2013)

Deep Learning Tutorial

http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf (visited 05.10.2017)

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho and Yoshua Bengio (2014)

On the Number of Linear Regions of Deep Neural Networks

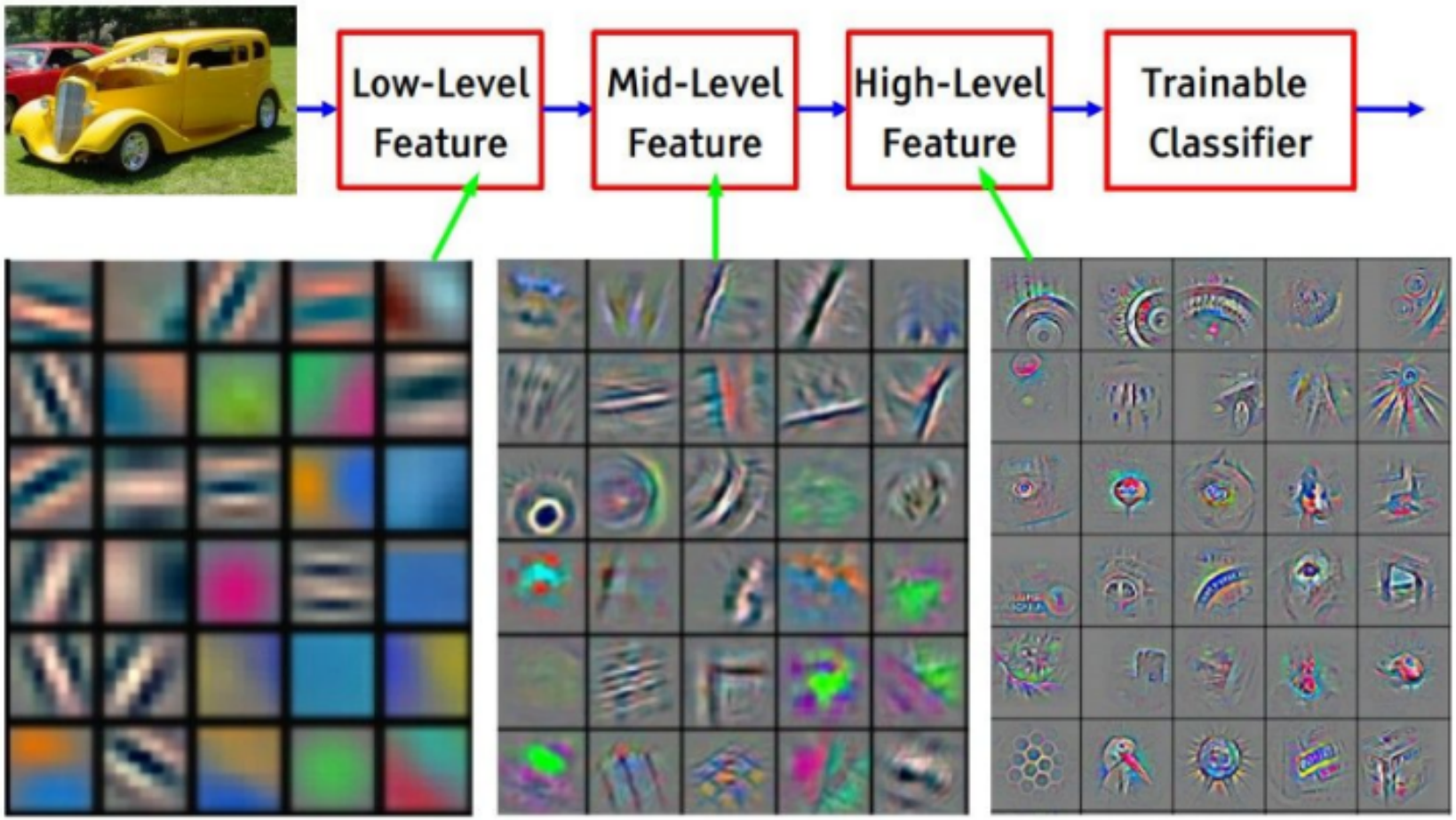
https://arxiv.org/pdf/1402.1869.pdf

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016)

Deep Learning

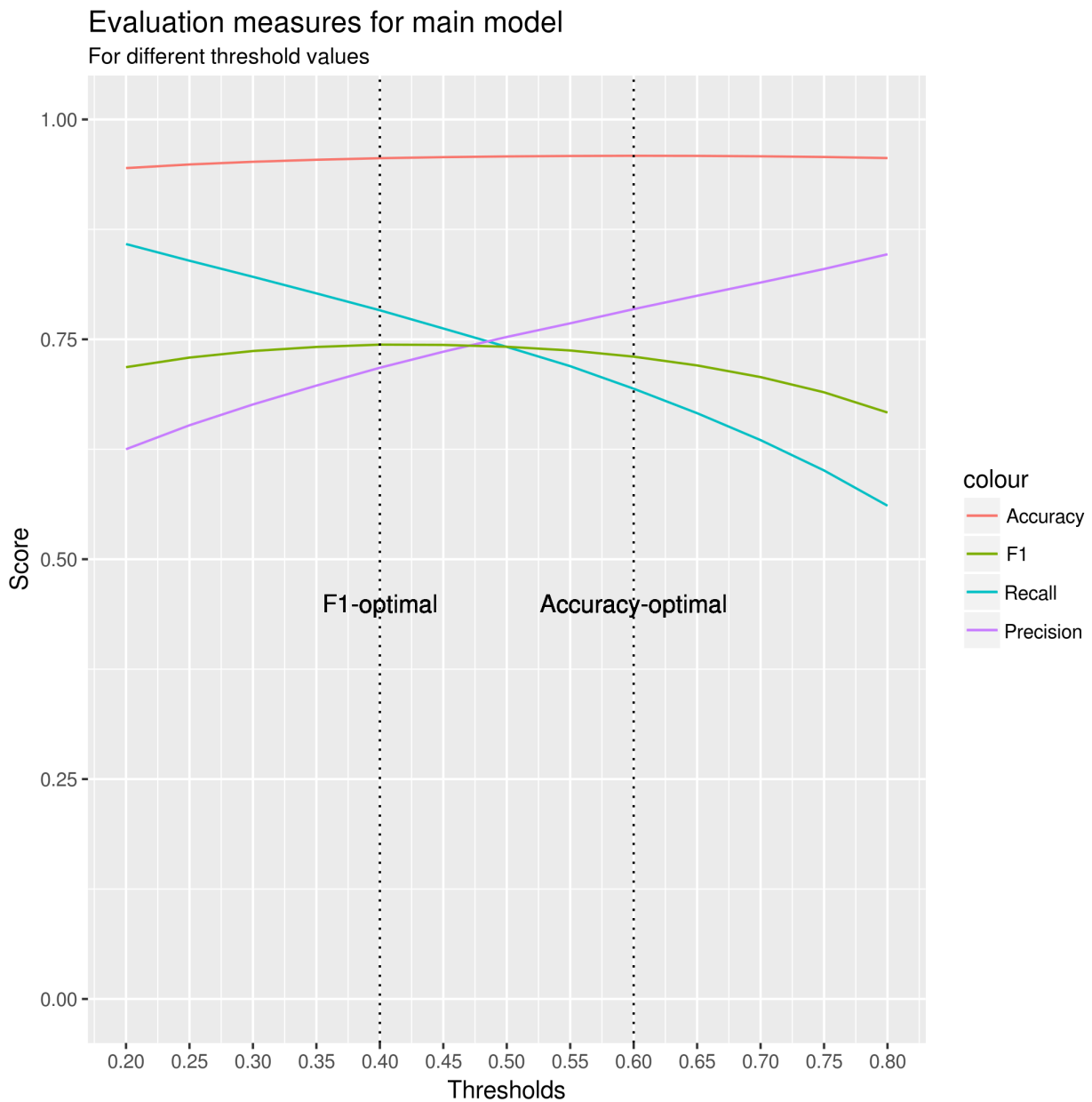
http://www.deeplearningbook.org/

Filter Visualization



Deep learning tutorial [LeCun & Ranzato, 2013]

Threshold evaluation



Training and hyperparameters

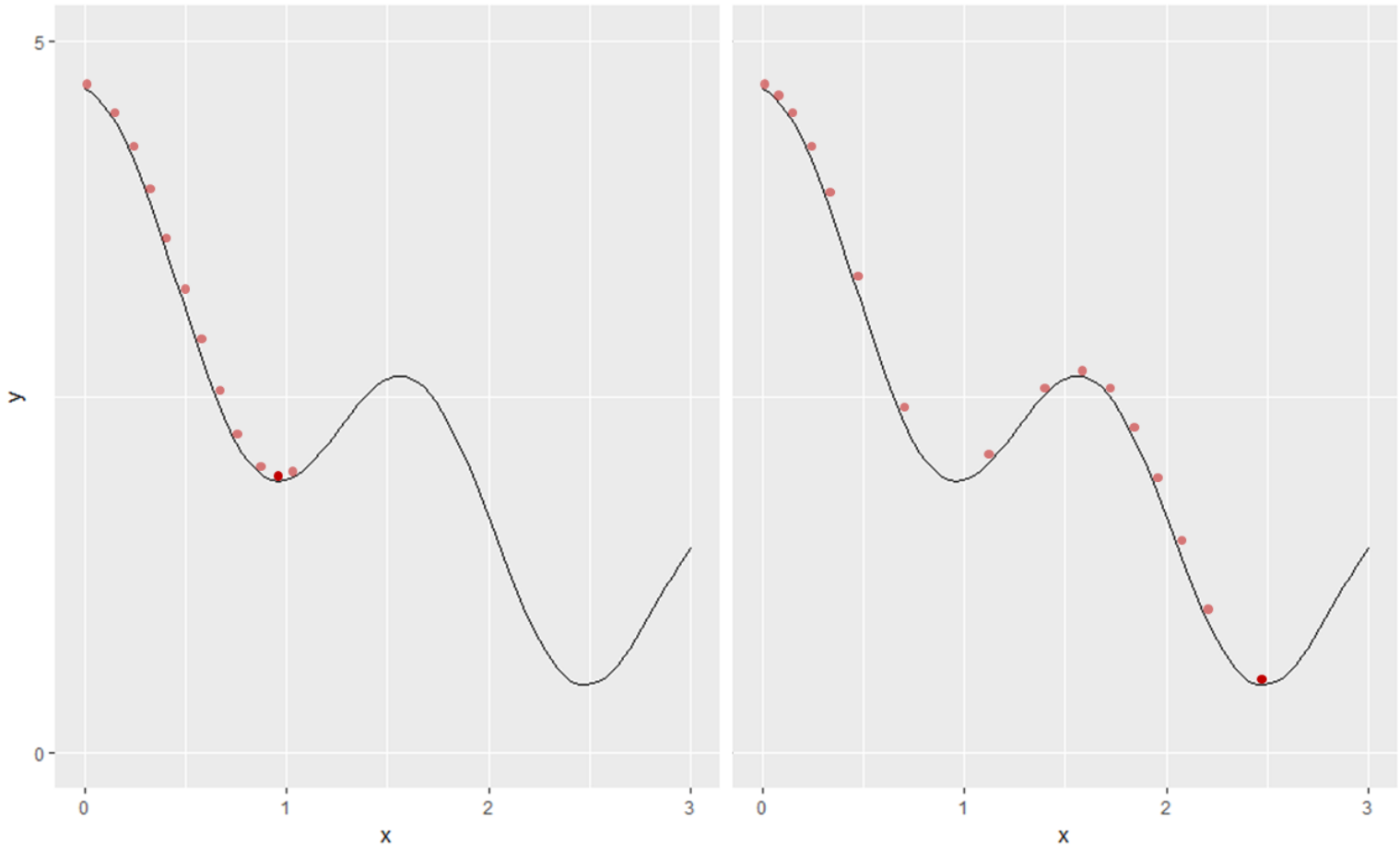
- Loss function: (average pixelwise) logistic loss/binary cross-entropy

$$L(y, f(x)) = -\frac{1}{512^2} \sum_{i=1}^{512} \sum_{j=1}^{512} \left[y_{(i,j)} log\ f(x_{(i,j)}) + (1 - y_{(i,j)}) log(1 - f(x_{(i,j)})) \right]$$

  where y corresponds to the ground truth and $f(x)$ to the networks prediction at pixel i, j.

- Optimizer: **Adam** (Adaptive Moment Estimation). Combination of:
  - AdaGrad: different learning rates for each parameter
  - and RMSProp: learning rates are adapted based on the average of recent magnitudes of the gradients

Momentum



The red dots represent the steps while optimizing the function.
Left: gradient descent, right: gradient descent with **momentum**.

Google maps policy

- Academic usage of the Google Maps is permitted according to the Google terms of use:

Proposed use	OK to use?	Additional information
Books	Yes	It's fine to use a handful of images, as long as you're not distributing more than 5,000 copies or using the Content in guidebooks.
Periodicals	Yes	This includes newspapers, magazines and journals.
Reports and presentations	Yes	This includes research papers, internal reports, presentations, proposals and other related professional documents.
Guidebooks	No	You may not use the Content as a core part of printed navigational material (for example, tour books).
Consumer goods	No	This includes retail products or retail product packaging (for example, t-shirts, beach towels, shower curtains, mugs, posters, stationery, etc.).

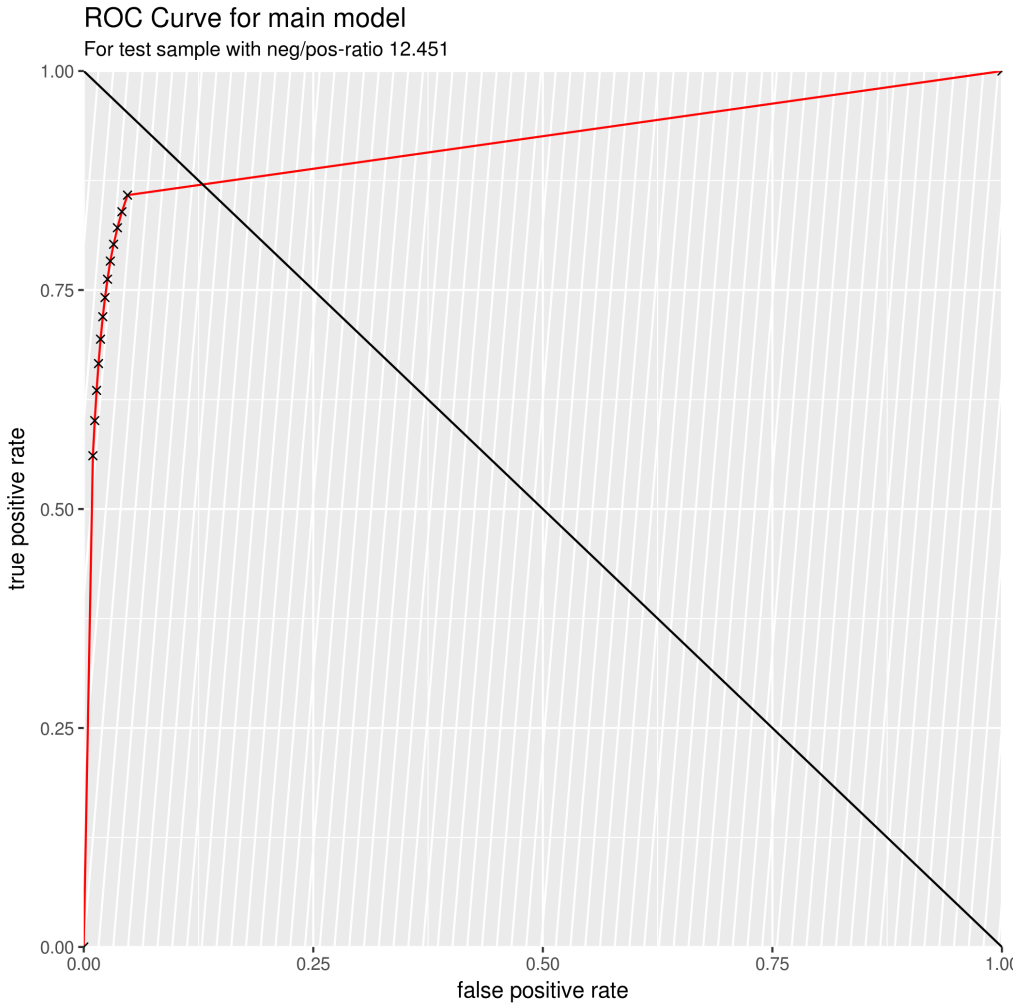
- Google offers premium plan with higher usage limits and resolution for professional usage

Additional information

- Resolution:
  - Zoom 18 in Google Maps correponds to resolution 1: 4513.988880
  - Resolution: 72 ppi (pixel per inch) $\rightarrow$ width of 1 pixel $=$ 0.035cm
  - 1 pixel on image corresponds to 157.990 cm in real

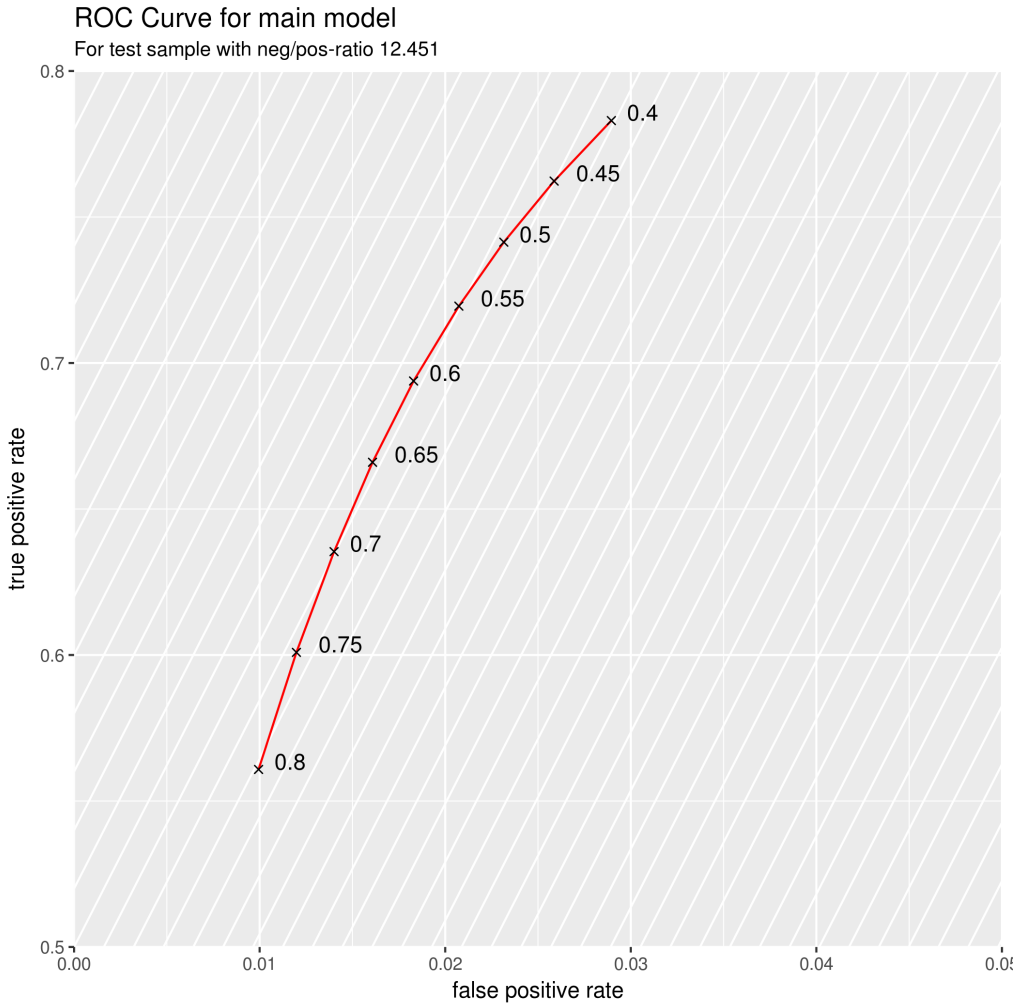
ROC analysis

- Method to detect accuracy-optimal threshold for scores



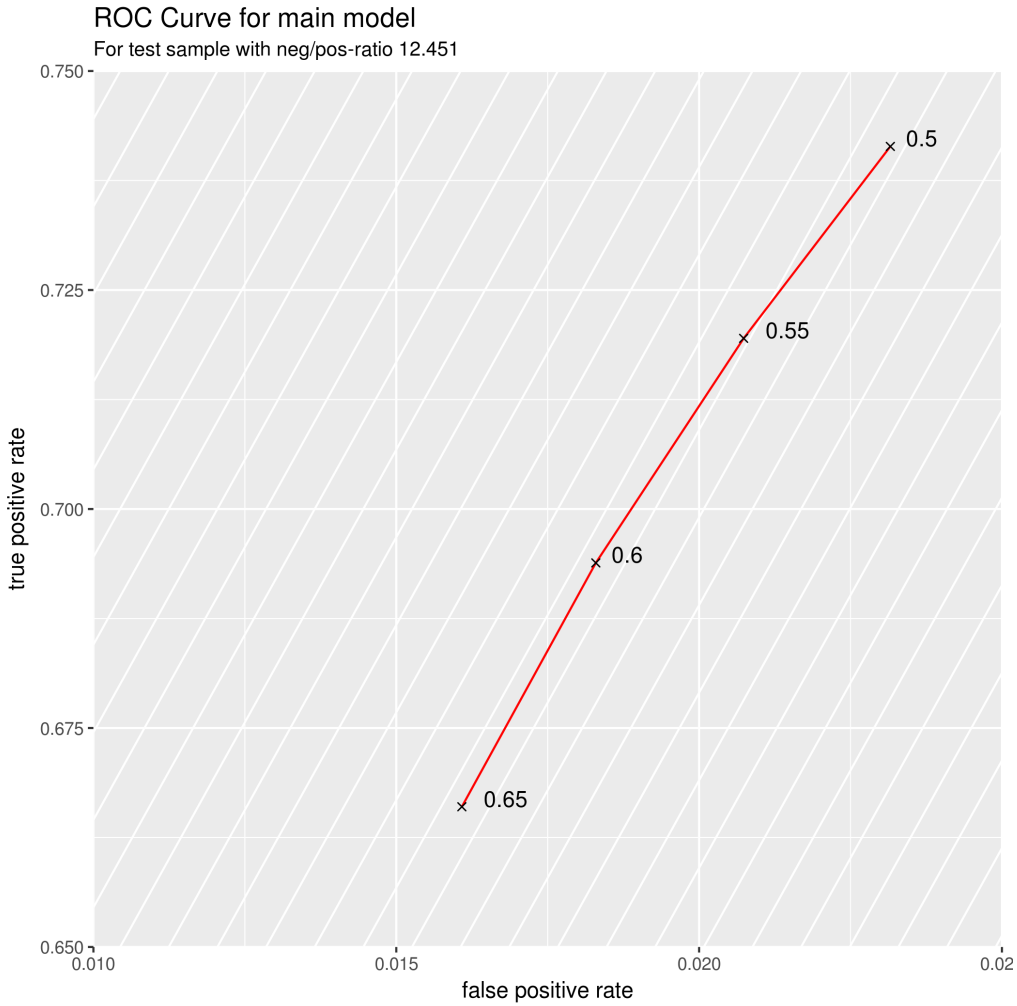
ROC analysis

- Method to detect accuracy-optimal threshold for scores



ROC analysis

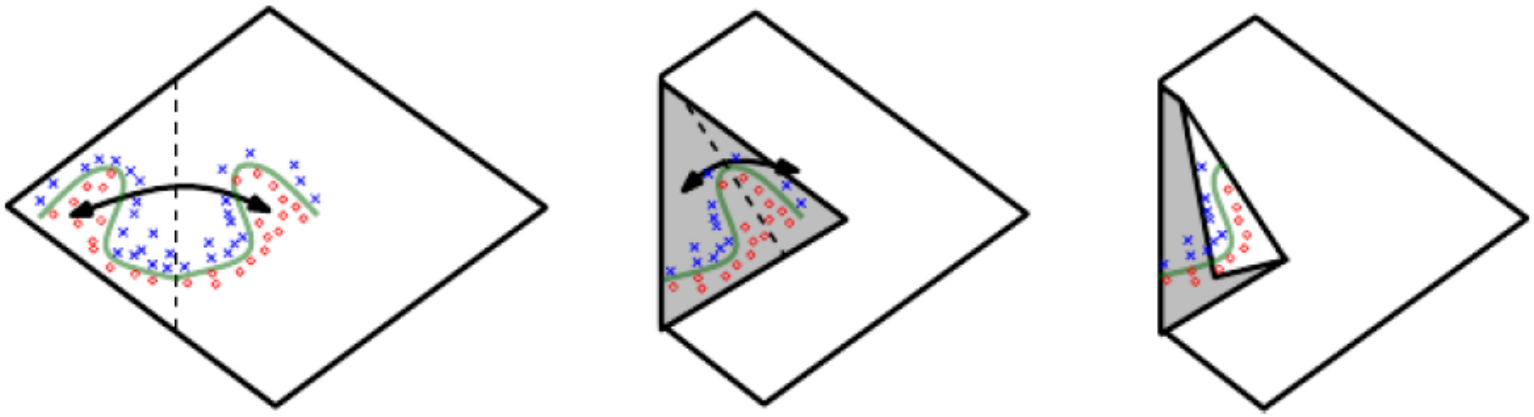
- Method to detect accuracy-optimal threshold for scores



- ROC-optimal threshold: 0.6

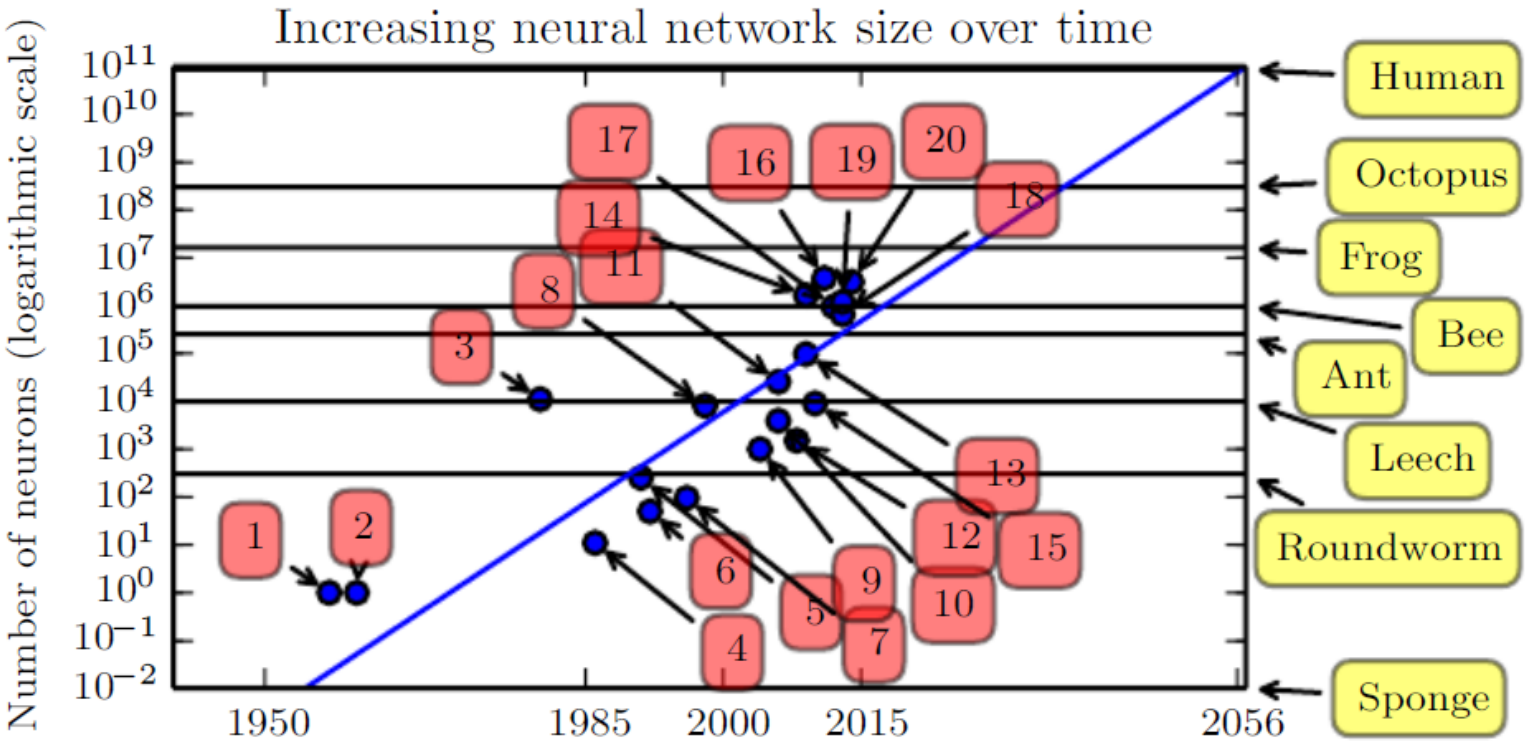
Why add more Layers?

- Each layer in a neural network adds its own degree of non-linearity to the model.



An intuitive, geometric explanation of the exponential advantage of deeper networks formally [Montúfar et al., 2014]

Network size over time



Network sizes over time. 1: Perceptron, 5: Recurrent neural network for speech recognition, 8: LeNet-5, 10: Deep belief network, 20: GoogLeNet. For more details, see: [Ian Goodfellow et al., 2016]