Deep Learning

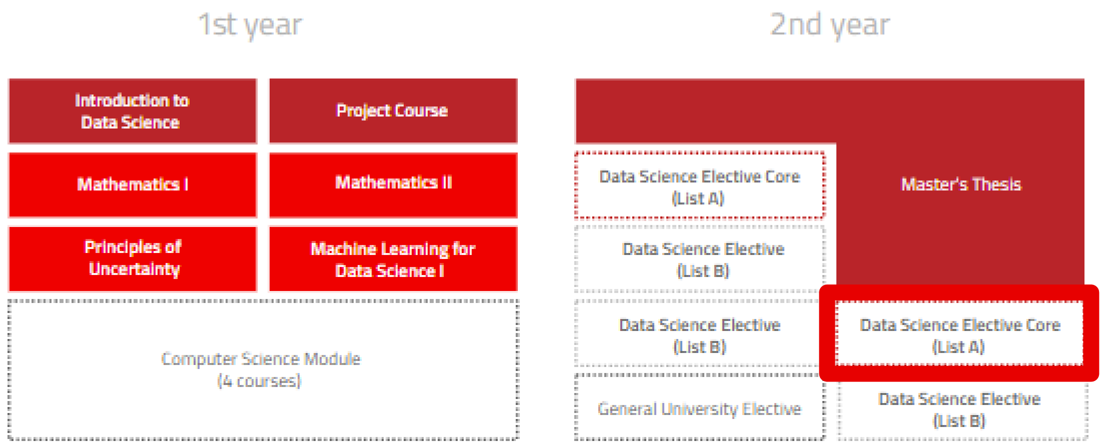
Introductory information

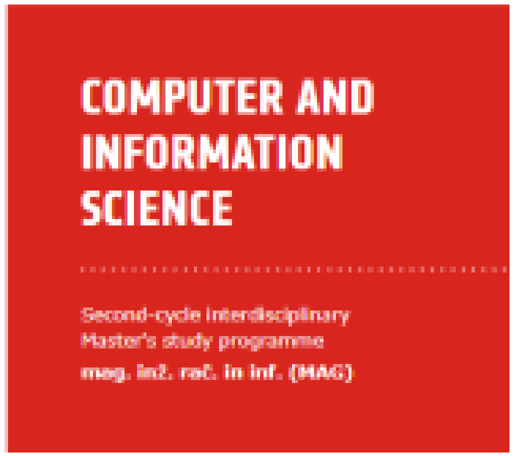
Danijel Skočaj
University of Ljubljana
Faculty of Computer and Information Science

Academic year: 2022/23

About the course

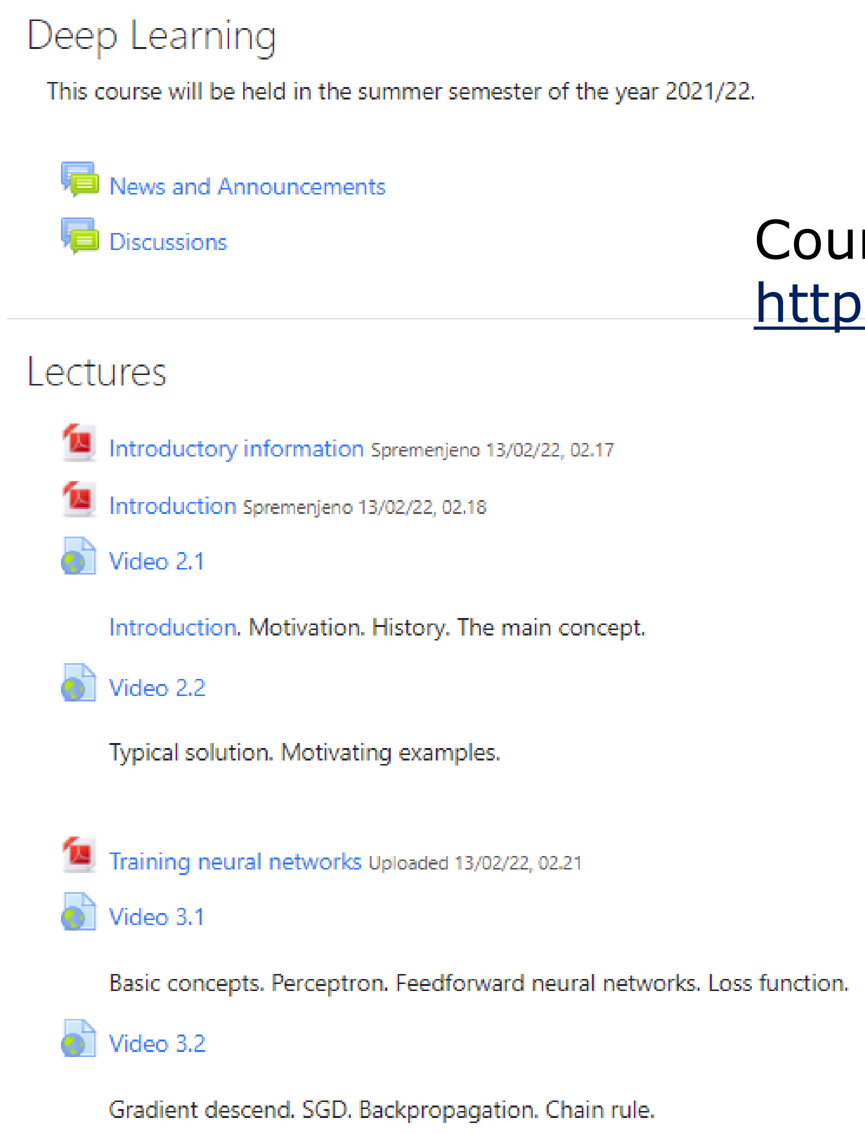
- **Deep Learning – DL**
- Data Science -Second-cycle Master's study programme track
  - data science elective core course
- Computer and Information Science, Second-cycle interdisciplinary Master's study programme
  - elective course
- 6 ECTS credits
- Lectures on Wednesday 11:15 - 14:00 in P21
- Tutorials on *
  - Mon. 15:15 - 17:00 in PR14
  - Tue. 9:15 - 11:00 in PR10
  - Fri. 16:15 – 18:00 in PR14


1st year
2nd year
Introduction to Data Science
Project Course
Mathematics I
Mathematics II
Principles of Uncertainty
Machine Learning for Data Science I
Computer Science Module (4 courses)
Master's Thesis
Data Science Elective Core (List A)
Data Science Elective (List B)
Data Science Elective (List B)
Data Science Elective Core (List A)
General University Elective
Data Science Elective (List B)


COMPUTER AND INFORMATION SCIENCE

Second-cycle interdisciplinary Master's study programme
mag. inž. rač. in inf. (MAG)

Course home page



Deep Learning

This course will be held in the summer semester of the year 2021/22.

News and Announcements

Discussions

Lectures

Introductory information Spremenjeno 13/02/22, 02.17

Introduction Spremenjeno 13/02/22, 02.18

Video 2.1

Introduction. Motivation. History. The main concept.

Video 2.2

Typical solution. Motivating examples.

Training neural networks Uploaded 13/02/22, 02.21

Video 3.1

Basic concepts. Perceptron. Feedforward neural networks. Loss function.

Video 3.2

Gradient descend. SGD. Backpropagation. Chain rule.

Course home page:
https://ucilnica.fri.uni-lj.si/course/view.php?id=368

Lecturer

- Prof. dr. Danijel Skočaj
  - Visual Cognitive Systems Laboratory
  - e-mail: danijel.skocaj@fri.uni-lj.si
  - url: http://www.vicos.si/danijels
  - tel: 01 479 8225
  - room: second floor, room R2.57
  - office hours: Tursdays, 13:00-14:00 (or at other times, send an email)

- Teaching assistant:
  As. Vitjan Zavrtanik
  - Visual Cognitive Systems Laboratory
  - e-mail: vitjan.zavrtanik@fri.uni-lj.si
  - tel: 01 479 8245
  - room: second floor, room R2.37

Course goals

Convolutional neural networks and related deep learning approaches have proven to be a very efficient way of finding the representations and building a classifier in a unified framework that yields excellent results in tasks in computer vision, NLP and other fields.

The main goal of this course is to introduce the students into the field of deep learning, to make them deeply understand the deep-learning principles as well to master the deep learning-related techniques for practical applications and get acquainted with the SOTA approaches.

Content

Theory

Applications

SOTA

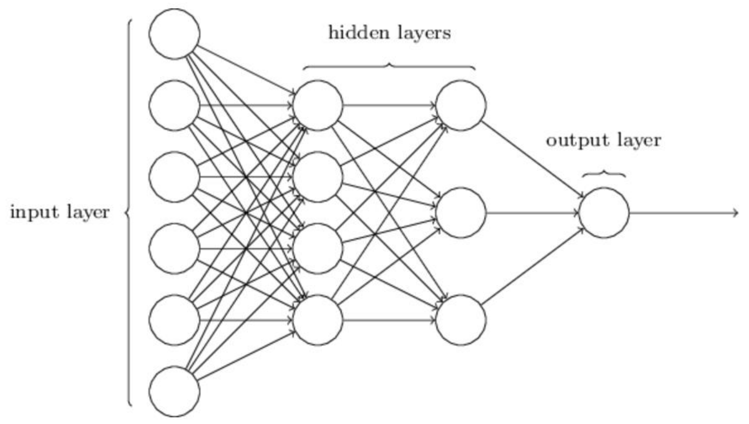
Details

Main features

Main idea

Content – Neural networks

- Basic concepts
  - Introduction and motivation
  - Brief history
  - Feedforward neural networks
- Training neural networks
  - Gradient Descend
  - Backpropagation
  - Loss and activation functions
- Optimizing training
  - Regularisation, normalisation, optimisers
  - Transfer learning



$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

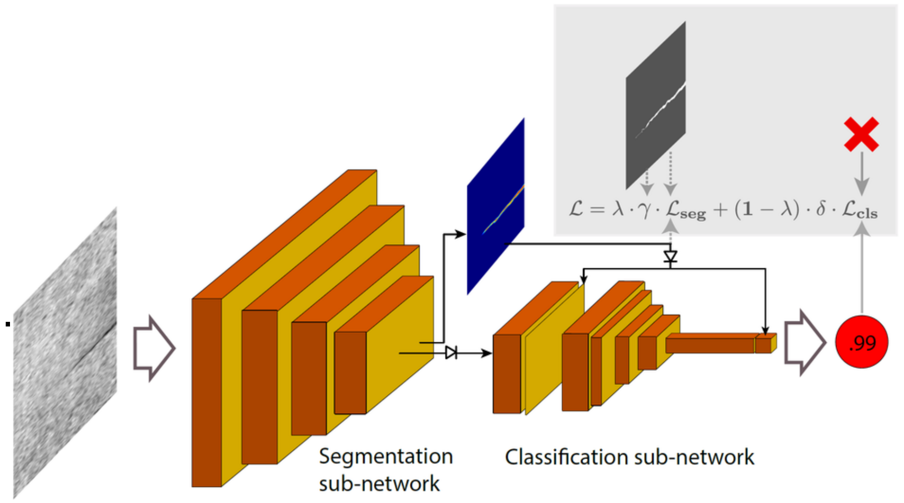
$$\frac{\partial C}{\partial b^l_j} = \delta^l_j$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$$

Basic concepts. Perceptron. Feedforward neural networks. Loss function. Gradient descend. SGD. Backpropagation. Chain rule. Equations of backpropagation. Backpropagation and SGD. Computational graph. Backprop summary. Backprop example. Quadratic loss function. Cross-entropy loss function. SoftMax. Categorical cross-entropy. Activation functions. Sigmoid. Tanh. ReLU. Overfitting. Regularisation. L2 and L1 regularisation. Regularised SGD. Dropout. Data augmentation. Pre-processing the data. Weight initialisation. Batch normalisation. Parameter-update optimisers. Momentum. Nesterov. Adagrad. RMSProp. Adam. Learning rate decay. Learning rate schedules. Setting up the network. Hyper-parameter optimisation.

Content – Convolutional neural networks

- Convolutional Neural Networks
  - Convolutional and other layers
  - Architectures of CNN
    - VGG, ResNet, MobileNet, EfficientNet, …
  - Building blocks
    - Inception, residual connection, separable convolution, …
  - Practical methodology
    - Data augmentation, transfer learning
  - Explainability of CNNs



$\mathcal{L} = \lambda \cdot \gamma \cdot \mathcal{L}_{\text{seg}} + (1-\lambda) \cdot \delta \cdot \mathcal{L}_{\text{cls}}$
.99
Segmentation sub-network
Classification sub-network

Convolutional neural networks. Convolution. Convolution layer. Sparse connectivity. Receptive field. Parameter sharing. Translation equivariance. CNNs as FC networks. CNN principle. Stride. Padding. Conv layer parameters. Executing convolution. Pooling layer. Fully connected layer. CNN architecture. Milestone CNN architectures. LeNet-5. AlexNet. VGGNet. GoogleLeNet. Inception. Auxiliary output. ResNet. Residual connection. Ensemble methods. SENet. Feature recalibration. Wide ResNet. ResNext. Parallel pathways. DenseNet. MobileNets. Depthwise separable convolution. Pointwise separable convolution. ShuffleNet. Group convolution. Neural architecture search. NASNet. EfficientNet. Architecture overview. Transfer learning. Data augmentation. AutoAugment. CNN regularisation. Cutout. Mixup. Explainability of CNNs. Visualising filters. Nearest neighbours. Dimensionality reduction. Visualising activations. Sensitivity to occlusions. LIME. Maximally Activating Patches. DeConvNet. Saliency maps via backprop. Guided backpropagation. Class Activation Maps. Grad-CAM. Adversarial images. FGSM and BIM. DeepFool. Accuracy-perturbation curves. Universal perturbations. SparseFool. Adversarial training.

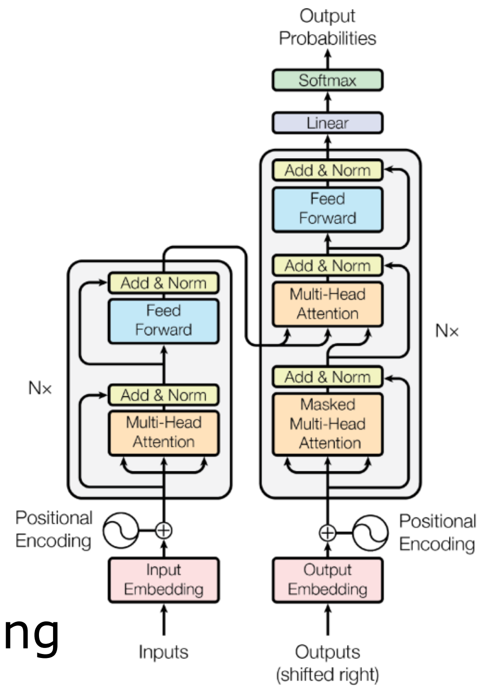
Content – Computer vision

- Computer vision
  - Classification, semantic segmentation
  - Detection, instance segmentation, panoptic segmentation
  - Transpose convolution, FPN, atrous convolution, unpooling…
  - Mask R-CNN, Yolo, RetinaNet, FCOS, U-Net, DeepLab, …
  - Applications in Computer Vision

Visual information. Computer vision tasks. Semantic segmentation. Fully-convolutional approach. Unpooling. Upsampling. Transpose convolution. FCN. Deconvolution network. SegNet. Encoder-decoder architecture. U-net. Shortcut connections. PSP-Net. Pyramid Pooling. DeepLab. Atrous convolution. Atrous Spatial Pyramid Pooling. Semantic segmentation overview. Beyond segmentation. Surface-defect detection. Counting. Obstacle detection. Image enhancement. Spatially Adaptive Filter Units. Semantic edge detection. Detection of single object. Bounding box regression. Detection of multiple objects. Two-stage object detectors. Region proposal methods. R-CNN. Fast R-CNN. RoI pooling. Faster RCNN. Region Proposal Network. Instance segmentation. Mask R-CNN. RoI Align. Feature Pyramid Network. Task specific heads. Human pose detection. Panoptic Feature Pyramid Networks. Single-stage object detectors. SSD. Anchors. Yolo. RetinaNet. Focal loss. FCOS. Traffic sign detection. Face detection. Object detection arhitectures overview. Performance of object detectors. Object detection performance measures. Semantic segmentation performance measures. etc.

Sequential data, Transformers and NLP

- Dealing with sequential data
  - Accounting for dependencies in time, Backpropagation through time
  - RNN, LSTM, GRU, Attention in RNNs
- Transformers
  - Transformers architecture
  - Self-attention, multihead attention
- Transformers in Natural language processing
  - BERT, RoBERTa, ALBERT, DeBERTa, GTP, GTP-2, GTP-3, ChatGPT, …
  - Unsupervised generative pretraining, supervised discriminative fine-tunning
  - Applications in NLP



Sequential data. CNN-based approach. Requirements. Sequential problems. Recurrent Neural network. Unfolding RNN. Reccurence formula. Computational graph. Character-level language model. Backpropagation through time. Predicting the next letter. Interpreting the hidden cell values. RNN trade-offs. Image captioning. Multilayer RNNs. RNN gradient flow. Long Short Term Memory. LSTM gradient flow. Gated Recurrent Units. RNN variants. Bidirectional LSTM. LSTM examples. Attention mechanism. Attention in RNNs. Image captioning with attention.

Transformer architecture. Encoder. Self-attention. Multi-head attention. Positional encoding. Decoder. Encoder-decoder attention. Masked self-attention. Decoding. Transformer-XL. BERT. BERT pretraining. BERT fine-tuning. RoBERTa. ALBERT. DeBERTa. BERT examples. GPT. Unsupervised generative pretraining. Supervised discriminative fine-tuning. GPT-2. GPT-3. Speech recognition. Music transformer. ChatGPT. etc.

Transformers in vision, Generative models, Recap

- Transformers in Computer vision
  - ViT, DeiT, MViT, Swin, DETR, UP-DETR, DINO, …
  - Applications in computer vision
- Generative models
  - AE, VAE
  - Generative Adversarial Networks
  - Normalizing Flows, Diffusion Models
- Deep learning recap

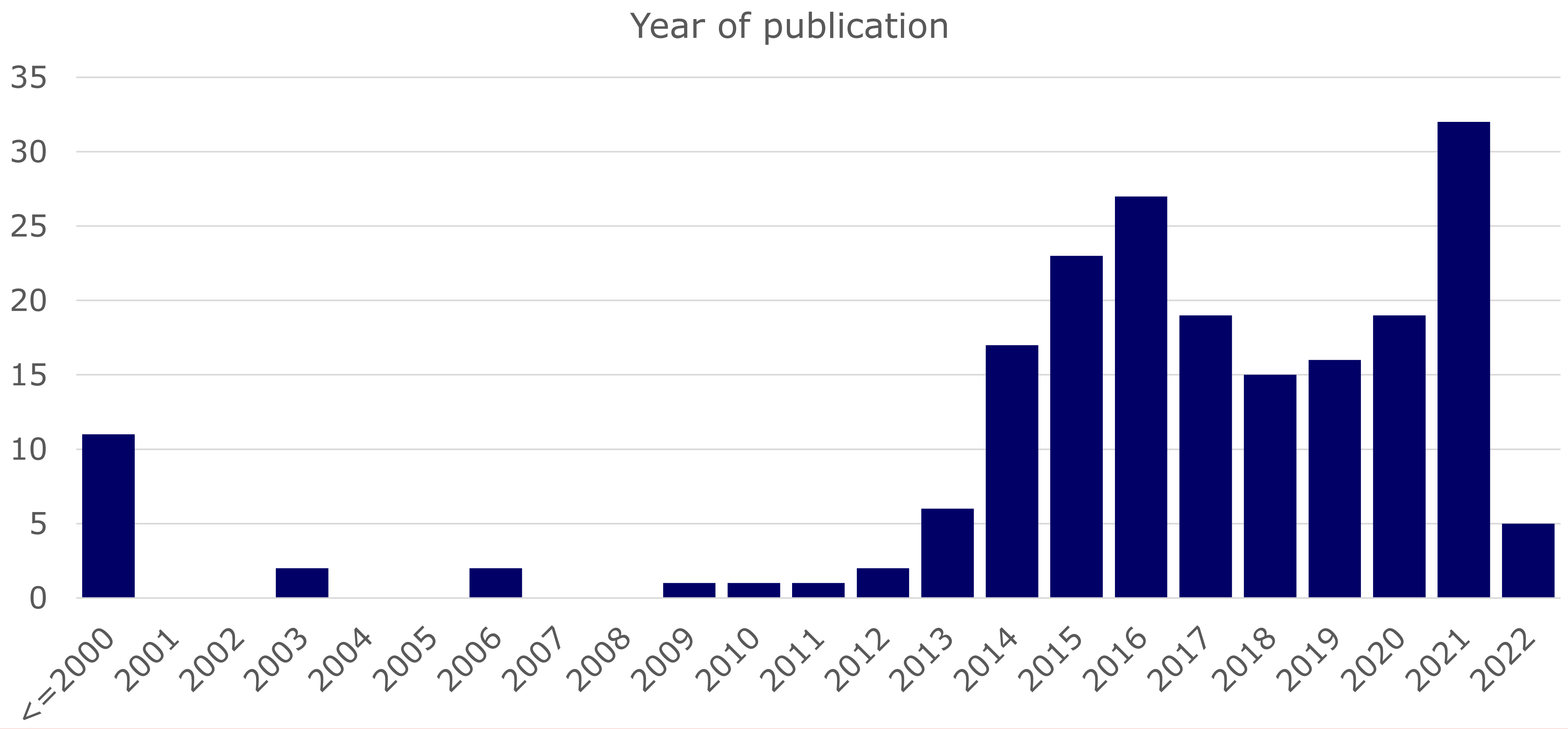
Image transformer. Vision transformer. DeiT. MViT. Swin transformer. Swin V2. SeMask. CvT. CoAtNet. DETR. Deformable DETR. UP-DETR. MaskFormer. Mask2Former. DINO. MLP-Mixer.

Gaussian and GMM. Principal Component Analysis. Robust projection. Autoencoders. Autoencoders for anomaly detection. Generative vs. discriminative approaches. Variational Autoencoders. VAE generating images. VAE in music. Generative Adversarial Nets. Training GAN. DCGAN. LSGAN. Wasserstain GAN. Improved Wasserstain GAN. Progressive GAN. Pix2pix cGAN. CycleGAN. BigGAN. SinGan. StyleGAN. F-AnoGAN. GANomaly. C-VTON. Normalizing flows. Real NVP. Flow++. Glow. DifferNet. FastFlow. PixelCNN. VQ-VAE. CLIP. DALL-E. Diffusion models. DDPM. GLIDE. DALL-E2. Stable diffusion.

Deep learning recap. Key factors for fair deep learning. Learning regimes. Deep reinforcement learning. RL for local navigation. Built in vs. Learned. Beyond CV and NLP. Function approximator. Development, deployment and maintenance. Problem solving. etc.

Research papers

- 199+ papers presented/mentioned
- All of them available online!
- Seminal papers and SOTA



Literature

- Michael A. Nielsen, Neural Networks and Deep learning, Determination Press, 2015
  http://neuralnetworksanddeeplearning.com/index.html

Neural Networks and Deep Learning

Nielsen, 2019

- Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016
  http://www.deeplearningbook.org/

Goodfellow et al., 2016

Deep Learning

An MIT Press book

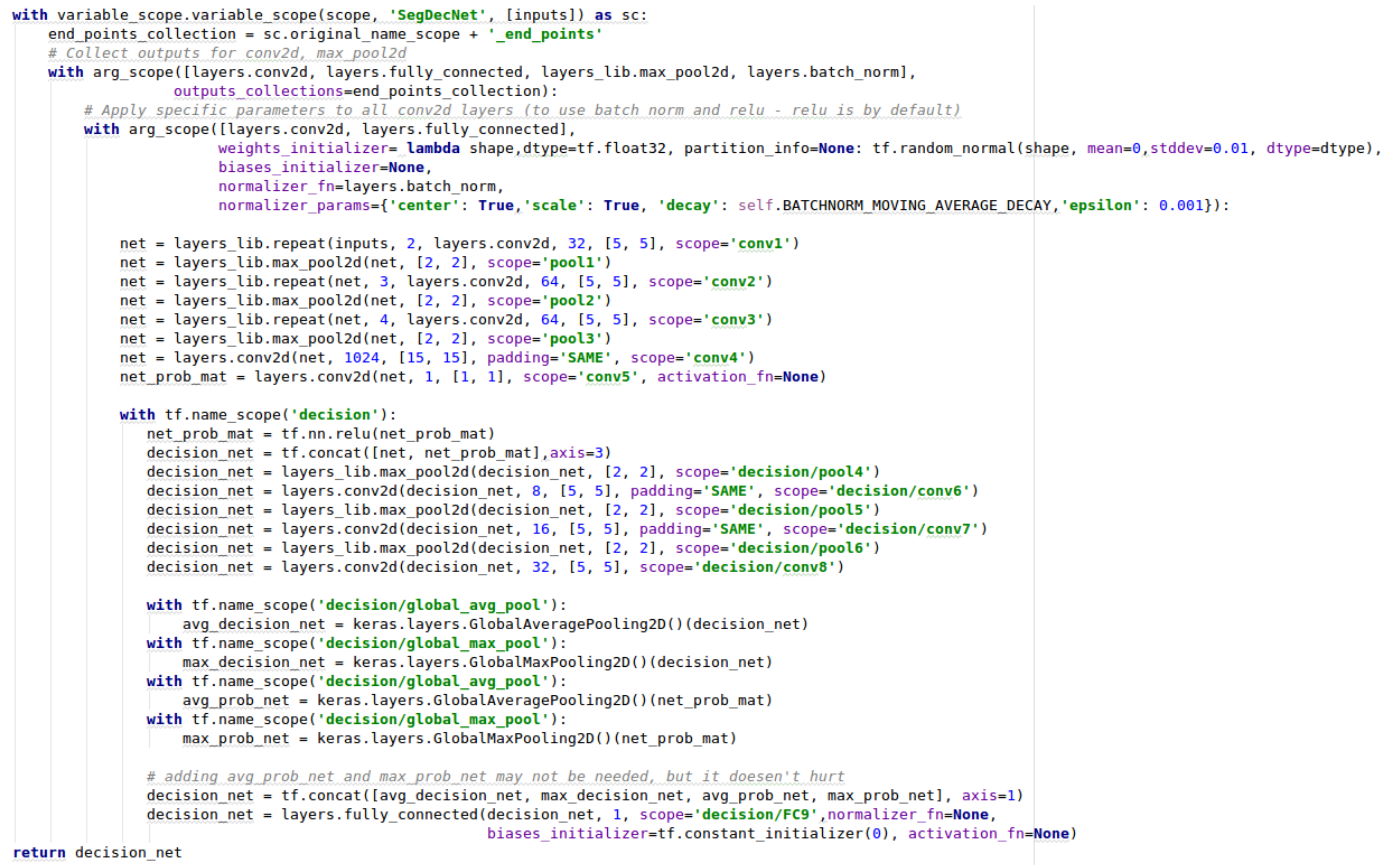
Ian Goodfellow and Yoshua Bengio and Aaron Courville

- Research papers!
- Other resources

Tutorials

- Lab exercises
- Consultations

- Mostly practically oriented
- Coding
- Examples
- Practical tips
- Programming advices
- Practical consultations
- Presentations of assignments
- Discussions on projects



```python
with variable_scope.variable_scope(scope, 'SegDecNet', [inputs]) as sc:
    end_points_collection = sc.original_name_scope + '_end_points'
    # Collect outputs for conv2d, max_pool2d
    with arg_scope([layers.conv2d, layers.fully_connected, layers_lib.max_pool2d, layers.batch_norm],
                   outputs_collections=end_points_collection):
        # Apply specific parameters to all conv2d layers (to use batch norm and relu - relu is by default)
        with arg_scope([layers.conv2d, layers.fully_connected],
                       weights_initializer= lambda shape,dtype=tf.float32, partition_info=None: tf.random_normal(shape, mean=0,stddev=0.01, dtype=dtype),
                       biases_initializer=None,
                       normalizer_fn=layers.batch_norm,
                       normalizer_params={'center': True,'scale': True, 'decay': self.BATCHNORM_MOVING_AVERAGE_DECAY,'epsilon': 0.001}):

            net = layers_lib.repeat(inputs, 2, layers.conv2d, 32, [5, 5], scope='conv1')
            net = layers_lib.max_pool2d(net, [2, 2], scope='pool1')
            net = layers_lib.repeat(net, 3, layers.conv2d, 64, [5, 5], scope='conv2')
            net = layers_lib.max_pool2d(net, [2, 2], scope='pool2')
            net = layers_lib.repeat(net, 4, layers.conv2d, 64, [5, 5], scope='conv3')
            net = layers_lib.max_pool2d(net, [2, 2], scope='pool3')
            net = layers.conv2d(net, 1024, [15, 15], padding='SAME', scope='conv4')
            net_prob_mat = layers.conv2d(net, 1, [1, 1], scope='conv5', activation_fn=None)

            with tf.name_scope('decision'):
                net_prob_mat = tf.nn.relu(net_prob_mat)
                decision_net = tf.concat([net, net_prob_mat],axis=3)
                decision_net = layers_lib.max_pool2d(decision_net, [2, 2], scope='decision/pool4')
                decision_net = layers.conv2d(decision_net, 8, [5, 5], padding='SAME', scope='decision/conv6')
                decision_net = layers_lib.max_pool2d(decision_net, [2, 2], scope='decision/pool5')
                decision_net = layers.conv2d(decision_net, 16, [5, 5], padding='SAME', scope='decision/conv7')
                decision_net = layers_lib.max_pool2d(decision_net, [2, 2], scope='decision/pool6')
                decision_net = layers.conv2d(decision_net, 32, [5, 5], scope='decision/conv8')

                with tf.name_scope('decision/global_avg_pool'):
                    avg_decision_net = keras.layers.GlobalAveragePooling2D()(decision_net)
                with tf.name_scope('decision/global_max_pool'):
                    max_decision_net = keras.layers.GlobalMaxPooling2D()(decision_net)
                with tf.name_scope('decision/global_avg_pool'):
                    avg_prob_net = keras.layers.GlobalAveragePooling2D()(net_prob_mat)
                with tf.name_scope('decision/global_max_pool'):
                    max_prob_net = keras.layers.GlobalMaxPooling2D()(net_prob_mat)

                # adding avg_prob_net and max_prob_net may not be needed, but it doesen't hurt
                decision_net = tf.concat([avg_decision_net, max_decision_net, avg_prob_net, max_prob_net], axis=1)
                decision_net = layers.fully_connected(decision_net, 1, scope='decision/FC9',normalizer_fn=None,
                                                      biases_initializer=tf.constant_initializer(0), activation_fn=None)
return decision_net
```

Software

- Basic neural networks in Python

- Convolutional neural networks using PyThorch, Transformers using HuggingFace,...

- Use your own favourite tool

Hardware

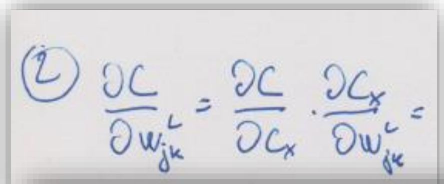
- Simple neural networks: any computer suffices
- CNNs and beyond: Nvidia GPU required (e.g., Nvidia Geforce GTX980 and above)

- Optionally use Google Colab
- Access to open research infrastructure HPC RIVR, NSC, ARNES

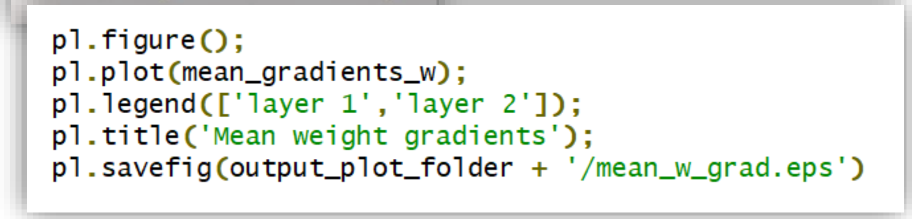
Requirements – practical part

- Assignments – homeworks
  - 4 assignments (in the first half of the semester)
    - theoretical, practical
  - Workload: app. 50 hours

- Project
  - On the topic of your choice
  - In the second half of the semester
  - Written report, paper-like
  - Workload: app. 50 hours



$$② \quad \frac{\partial C}{\partial w_{jk}^{L}} = \frac{\partial C}{\partial C_x} \cdot \frac{\partial C_x}{\partial w_{jk}^{L}} =$$



```
pl.figure();
pl.plot(mean_gradients_w);
pl.legend(['layer 1','layer 2']);
pl.title('Mean weight gradients');
pl.savefig(output_plot_folder + '/mean_w_grad.eps')
```

Deep Learning for Computer Vision
(Final project report)

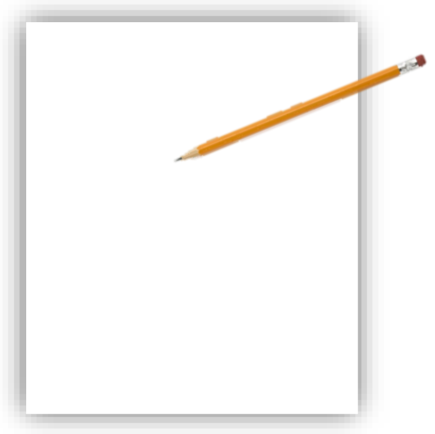
[illegible]

1 Introduction

[illegible]

Requirements – theoretical part

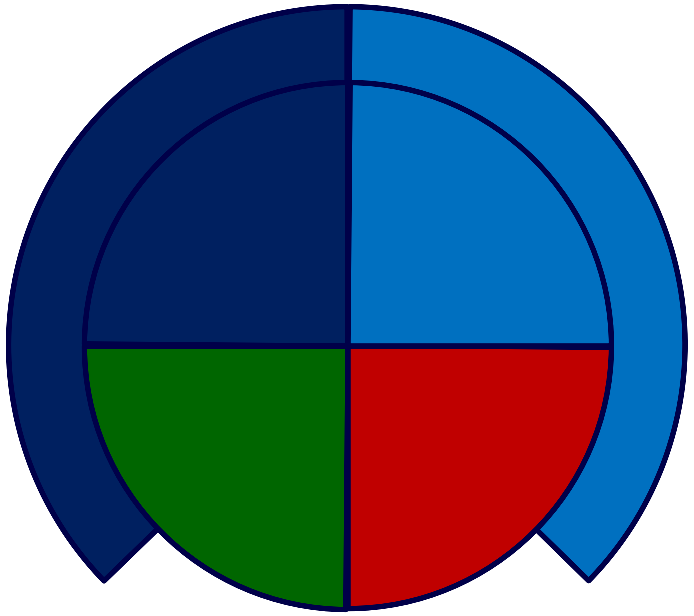
- Written exam
  - Rather short
  - Evaluating understanding of the underlying concepts
  - Derivating, calculating, describing
- Oral exam
  - Rather short
  - Questions about the main concepts and features
  - Discussion about assignments and project
  - Verifying understanding of the underlying concepts



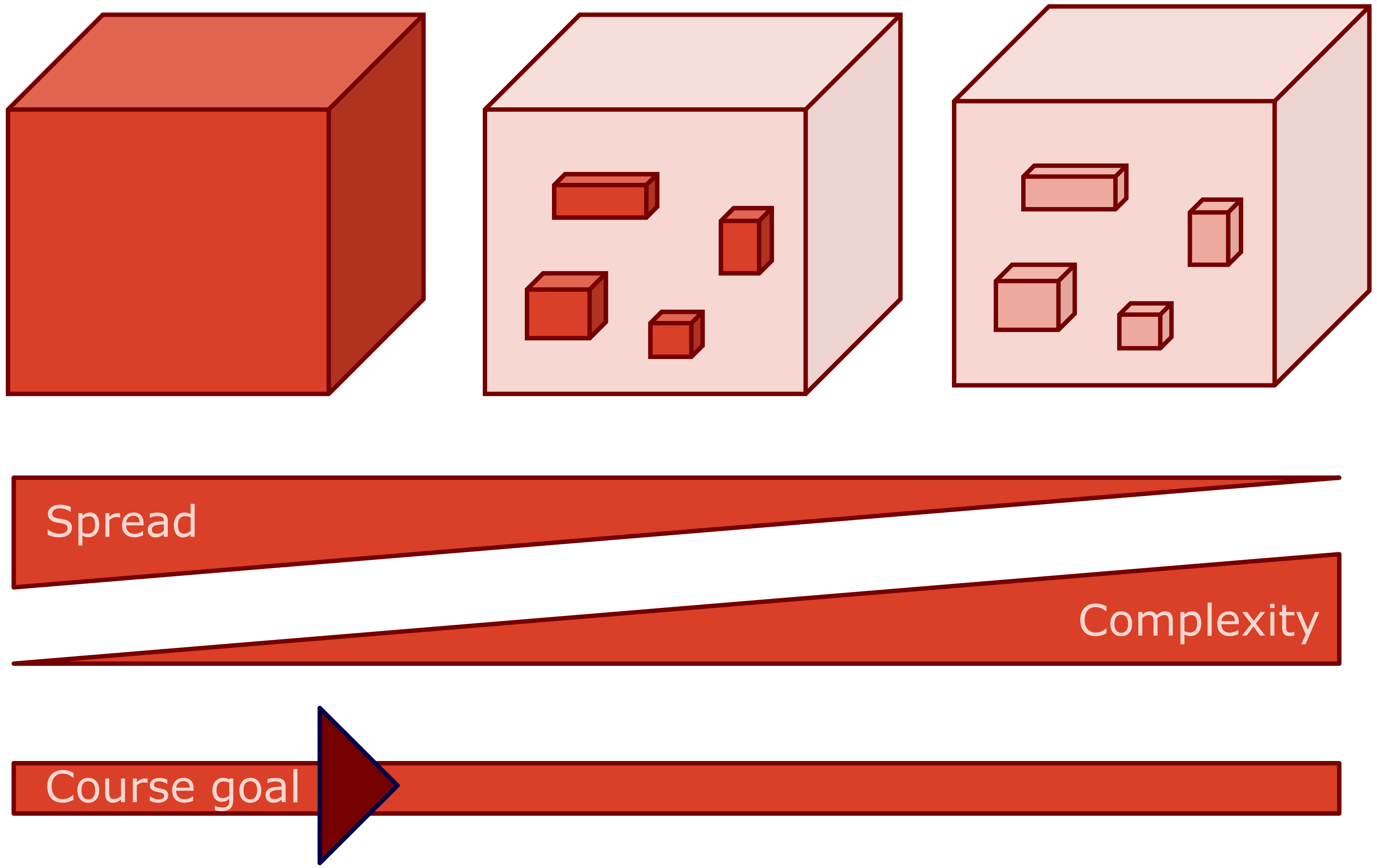
Grading

- Assignments: 25 pts
- Project: 25 pts
- Written exam: 25 pts
- Oral exam: 25 pts

- The student has to collect at least 50% in every task



Using deep learning


Spread
Complexity
Course goal