A Shallow Introduction into the Deep Machine Learning

Jan Čech
What is the “Deep Learning”? 

- Deep learning (by G. Hinton, DL pioneer, Turing Award 2018 holder) = both the classifiers and the features are learned automatically 

  ![Diagram](image)

  - Typically not feasible, due to high dimensionality 
  - Suboptimal, requires expert knowledge, works in specific domain only 

- Deep neural network (feature hierarchies) 

  ![Diagram](image)
What is the “Deep Learning”? Other definitions…

- **Andrew Ng** (founder of Google Brain, chief of Baidu AI research)
  - “**Very large neural networks** we can now have and … huge amounts of data that we have access to.”

- **Jeff Dean** (head of Google AI)
  - “When you hear the term deep learning, just think of a **large deep neural net**. Deep refers to the number of layers typically and so this kind of the popular term that’s been adopted in the press. I think of them as deep neural networks generally.”

- **Yoshua Bengio** (DL pioneer, Turing Award Holder 2018)
  - “Deep learning algorithms seek to exploit the unknown **structure** in the input distribution in order to discover good representations, **often at multiple levels**, with higher-level learned features defined in terms of lower-level features.”

- **Yann LeCun** (DL pioneer, Turing Award Holder 2018)
  - “Deep learning [is] … **a pipeline of modules all of which are trainable**. … deep because [has] multiple stages in the process of recognizing an object and all of those stages are part of the training.”
Deep Learning omnipresent

- Besides the Computer Vision DL is extremely successful in, e.g.
  - Automatic Speech Recognition
    - Speech to text, Speaker recognition
  - Natural Language Processing
    - Machine translation, Question answering
  - Robotics / Autonomous driving
    - Reinforcement learning
  - Data Science / Bioinformatics

- Shift of paradigm in Computer Vision
  - Large-scale image category recognition (ILSVRC’ 2012 challenge)

  INRIA/Xerox 33%,
  Uni Amsterdam 30%,
  Uni Oxford 27%,
  Uni Tokyo 26%,
  **Uni Toronto 16% (deep neural network)** [Krizhevsky-NIPS-2012]
Explosion of interest in “Deep Learning” after 2012

- Paper title keywords, CVPR 2019

- Number of attendees/submissions in major Computer Vision and Machine Learning grows exponentially

Data Source: https://hai.stanford.edu/
Examples of Deep learning in Computer Vision

- Image classification [Krizhevsky-NIPS-2012]
  - Input: RGB-image
  - Output: Single label (Probability Distribution over Classes)

- ImageNet dataset (14M images, 21k classes, Labels by Amazon Mechanical Turk)
- ImageNet Benchmark (1000 classes, 1M training images)
Examples of Deep learning in Computer Vision

- **Object Detection**
  - Multiple objects in the image [**RCNN**,**YOLO**,...]
  - E.g. Face [**Hu-Ramanan-2017**], Text localization [**Busta-2017**]
Examples of Deep learning in Computer Vision

- (3D) Pose estimation
  - [Hu-2018], [OpenPose]
Examples of Deep learning in Computer Vision

- Image Segmentation (Semantic/Instance Segmentation)
  - Each pixel has a label \([\text{Long-2015}], [\text{Mask-RCNN-2017}]\)
Examples of Deep learning in Computer Vision

- **Motion**
  - Tracking
  - Optical Flow [Neoral-2018]
    - Predict pixel level displacements between consecutive frames
Examples of Deep learning in Computer Vision

- Stereo (depth from two images)
- Depth from a single (monocular) image [Godard-2017]
Examples of Deep learning in Computer Vision

- Faces
  - Recognition / Verification
  - Gender/Age
  - Landmarks, pose
  - Expression, emotions

...already in commerce
Examples of Deep learning in Computer Vision

- Lip reading [Chung-2017]
Examples of Deep Learning in Computer Vision

- **Image-to-Image translation** [Isola-2017]
  - Day to Night
  - BW to Color

- **Deblurring, Super-resolution** [Šubrtová-2018]
Examples of Deep learning in Computer Vision

- Generative models
  - Generating photo-realistic samples from image distributions
  - Variational Autoencoders, GANs [Nvidia-GAN]

(Images synthetized by a random sampling)
Examples of Deep learning in Computer Vision

- Action/Activity recognition
- Neural Style Transfer
- Image Captioning
- and many more…

https://cs.stanford.edu/people/karpathy/deepimagesent/generationdemo/
History: (Artificial) Neural Networks

- Neural networks are here for more than 50 years
  - Rosenblatt-1956 (perceptron)
  - Minsky-1969 (xor issue, => skepticism)
Rumelhart and McClelland – 1986:
- Multi-layer perceptron,
- Back-propagation (supervised training)
  - Differentiable activation function
  - Stochastic gradient descent

Empirical risk

\[ Q(w) = \sum_{i=1}^{n} Q_i(w), \]

Update weights:

\[ w := w - \alpha \nabla Q_i(w). \]

What happens if a network is deep? (it has many layers)
What was wrong with back propagation?

- Local optimization only (needs a good initialization, or re-initialization)
- Prone to over-fitting
  - too many parameters to estimate
  - too few labeled examples
- Computationally intensive

=> Skepticism: A deep network often performed worse than a shallow one
Why does it work now?

- However nowadays:
  - Large collections of labeled data available
    - ImageNet (14M images, 21k classes, hand-labeled)
  - Reducing the number of parameters by weight sharing
    - **Convolutional** layers – [LeCun-1989]
  - Novel tricks to prevent overfitting of deep nets
  - Fast enough computers (parallel hardware, GPU)

=> Optimism: It works!
Computational power

GigaFLOPs per Dollar

Deep Learning Explosion

GTX 1080 Ti

GeForce GTX 580 (AlexNet)

GeForce 8800 GTX
Deep convolutional neural networks

- An example for Large Scale Classification Problem:
    - Recognizes 1000 categories from ImageNet
    - Outperforms state-of-the-art by significant margin (ILSVRC 2012)

```
<table>
<thead>
<tr>
<th>Stride of 4</th>
<th>96 Max pooling</th>
<th>Max pooling</th>
<th>Max pooling</th>
<th>Dense</th>
<th>Dense</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>55</td>
<td>27</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>11 224</td>
<td>55</td>
<td>256</td>
<td>384</td>
<td></td>
<td></td>
<td>4096</td>
</tr>
<tr>
<td>224</td>
<td>55</td>
<td>256</td>
<td>384</td>
<td>13</td>
<td>13</td>
<td>4096</td>
</tr>
</tbody>
</table>
```

“Alex-Net”

- 5 convolutional layers, 3 fully connected layers
- 60M parameters, trained on 1.2M images (~1000 examples for each category)
- Cross-Entropy loss (softmax log-loss)
Deep CNNs – basic building blogs

- A computational graph (chain/directed acyclic graph) connecting layers
  - Each layer has: Forward pass, Backward pass
  - The graph is end-to-end differentiable

1. Input Layer
2. Intermediate Layers
   1. Convolutions
   2. Max-pooling
   3. Activations
3. Output Layer
4. Loss function over the output layer for training
**Convolutional layer**

- **Input**: tensor \((W \times H \times D)\)
  - “image” of size \(W \times H\) with \(D\) channels

- **Output**: tensor \((W' \times H' \times D')\)

- A bank of \(D'\) filters of size \((K \times K \times D)\) is convolved with the input to produce the output tensor
  - Zero Padding \((P)\), extends the input by zeros
  - Stride \((S)\), mask shifts by more than 1 pixel
  - \(K \times K \times D \times D'\) parameters to be learned
Max-pooling layer

- Same inputs \((W \times H \times D)\) and outputs \((W' \times H' \times D)\) as convolutional layer
- Same parameters: Mask Size \((K)\), Padding \((P)\), Stride \((S)\)
- Same sliding window as in convolution, but instead of the dot product, pick maximum
- Non-linear operation
- No parameters to be learned

![Max-pooling diagram]
Activation functions

- Non-linearity, applied to every single cell of the tensor
- Input tensor and output tensor of the same size

- **Sigmoid**
  \[ \sigma(x) = \frac{1}{1+e^{-x}} \]

- **tanh**
  \[ \tanh(x) \]

- **ReLU**
  \[ \max(0, x) \]

- **Leaky ReLU**
  \[ \max(0.1x, x) \]

- **Maxout**
  \[ \max(w_1^T x + b_1, w_2^T x + b_2) \]

- **ELU**
  \[ \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases} \]

- ReLU is the simplest (used in the AlexNet, good baseline)
- Saturating non-linearity (sigmoid, tanh) causes “vanishing” gradient
Dee Convolutional neural networks

- Additional tricks: “Devil is in the details”
  - Rectified linear units instead of standard sigmoid
    => Mitigate vanishing gradient problem
  - Convolutional layers followed by max-pooling
    • Local maxima selection in overlapping windows (subsampling)
    => dimensionality reduction, shift insensitivity
  - Dropout
    • 50% of hidden units are randomly omitted during the training, but weights are shared in test time
    • Averaging results of many independent models (similar idea as in Random forests)
    => Probably very significant to reduce overfitting
  - Data augmentation
    • Images are artificially shifted and mirrored (10 times more images)
    => transformation invariance, reduce overfitting
Deep convolutional neural networks

- **Supervised training**
  - The training is done by a standard back-propagation
  - enough labeled data: 1.2M labeled training images for 1k categories
  - Learned filters in the first layer
    • Resemble cells in primary visual cortex

  ![Learned first-layer filters](image)

- **Training time:**
  - 5 days on NVIDIA GTX 580, 3GB memory (Krizhevsky, today faster)
  - 90 cycles through the training set

- **Test time (forward step) on GPU**
  - 5 ms/image in a batch mode
Early experiments 1: Category recognition

  - Network pre-trained for 1000 categories provided
- Which categories are pre-trained?
  - 1000 “most popular” (probably mostly populated)
  - Typically very fine categories (dog breeds, plants, vehicles…)
  - Category “person” (or derived) is missing
  - Recognition accuracy subjectively surprisingly good…
It is not a texture only...

tiger, Panthera tigris

tiger cat

tabby, tabby cat

lynx, catamount

jaguar, panther, Panthera onca, Felis onca

Saint Bernard, St Bernard

Welsh springer spaniel

Blenheim spaniel

Irish setter, red setter

Leonberg
Early experiments 2: Category retrieval

- 50k randomly selected images from Profimedia dataset
- Category: Restaurant (results out of 50k-random-Profiset)
Early experiments 2: Category retrieval

- Category: stethoscope (results out of 50k-random-Profiset)
Early experiments 3: Similarity search

- Indications in the literature that the last hidden layer carry semantics
  - Last hidden layer (4096-dim vector), final layer category responses (1000-dim vector)
  - New (unseen) categories can be learned by training (a linear) classifier on top of the last hidden layer
    - Oquab, Bottou, Laptev, Sivic, CVPR, 2014
    - Girshick, Dphanue, Darell, Malik, CVPR, 2014
  - Responses of the last hidden layer can be used as a compact global image descriptor
    - Semantically similar images should have small Euclidean distance
Early experiments 3: Similarity search

- Qualitative comparison: (20 most similar images to a query image)
     - Nearest neighbour search in 20M images of Profimedia
     - Standard global image statistics (e.g. color histograms, gradient histograms, etc.)
  2. Caffe NN (last hidden layer response + Euclidean distance),
     - Nearest neighbour search in 50k images of Profimedia
     - 400 times smaller dataset!

MUFIN results
Early experiments 3: Similarity search

Caffe NN results

1: 0
2: 6177.14
3: 6700.79
4: 6720.73
5: 6802.73

6: 5870.66
7: 6873.84
8: 6969.95
9: 7263.94
10: 7254.6

11: 7261.05
12: 7278.5
13: 7399.02
14: 7448.54
15: 7454.2

16: 7475.14
17: 7516.24
18: 7529.46
19: 7539.31
20: 7570.21
Early experiments 3: Similarity search

MUFIN results
Early experiments 3: Similarity search
Early experiments 3: Similarity search

MUFIN results
Early experiments 3: Similarity search

Caffe NN results

1: 0
2: 2812.02
3: 2968.18
4: 3189.3
5: 3284.86
6: 3286.28
7: 3304.93
8: 3402.86
9: 3433.69
10: 3473.81
11: 3495.67
12: 3528.47
13: 3549.56
14: 3559.5
15: 3562.74
16: 3574.01
17: 3576.81
18: 3597.88
19: 3599.39
20: 3662.85
Early experiments 3: Similarity search

MUFIN results
Early experiments 3: Similarity search
Early experiments 3: Similarity search

MUFIN results
Early experiments 3: Similarity search

Caffe NN results
Novel tricks

- Network initialization
  - Mishkin, Matas. *All you need is a good init.* ICLR 2016
  - Weights initialization: zero mean, unit variance, orthogonality

- Batch normalization
  - Zero mean and unit variance weights are “supported” during training to avoid vanishing gradient

⇒ Small sensitivity to learning rate setting (can be higher, faster training – 10 times fewer epochs needed)
⇒ Regularizer (dropout can be excluded/lower) (better optimum found)
Novel tricks II.

- Exponential Linear Units (ELU) [Clevert et al., ICLR 2016]
  - Self normalizing properties, batch normalization unnecessary
  - Faster training reported

- ADAM optimizer [Kingma and Ba, ICLR 2015]
  - (ADAdptive Moments)
  - Often improves over SGD (with momentum),
  - Low sensitivity on learning rate setting

$$f(x) = \begin{cases} 
  x & \text{if } x > 0 \\
  \alpha (\exp(x) - 1) & \text{if } x \leq 0 
\end{cases}$$
Novel architectures

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

HoG + DPM ➔ CNN

AlexNet
VGG Net
GoogLeNet
ResNet

classification error %

28.2 25.8 16.4 11.7 7.3 6.7 3.57 5.1

number of layers


=> “Go deeper”
CNN architectures

- **AlexNet**
  - [Krizhevsky et al., NIPS 2012]

```
11x11 conv, 96, /4, pool/2
5x5 conv, 256, pool/2
3x3 conv, 384
3x3 conv, 384
3x3 conv, 256, pool/2
fc, 4096
fc, 4096
fc, 1000
```
CNN architectures

- VGG Net: VGG-16, VGG-19
  - [Simonyan and Zisserman, ICLR 2015]
  - Deeper than AlexNet
  - Smaller filters (3x3 convolutions), more layers
    => Same effective receptive field,
    but more “non-linearity”
CNN architectures

- **GoogLeNet**
  - [Szegedy et al., CVPR 2015]
  - 22 layers, No Fully-Connected layers
  - Accurate, much less parameters
  - “Inception” module (Net in net)
CNN architectures

- **ResNet**
  - [He et al., CVPR 2016]
  - Residual modules, 152 layers

=> Plain deeper models are not better (vanishing gradient)

- Residual modules, 152 layers

\[ H(x) = F(x) + x \]
CNN architectures

- **ResNeXt**
  - [Xie-CVPR-2018]
  - Improvement of ResNet
  - Cardinality
    - number of branches in a block
  - “Increasing cardinality, better than going wider or deeper"
CNN architectures

- DenseNet
  - [Huang-CVPR-2017]
  - Densifying Skip connections
  - Chain of several “dense blocks”
  - Argument: Features are reused
  - Higher accuracy with fewer parameters over ResNet reported
  - Best paper award @ CVPR
CNN architectures

- Squeeze-and-Excitation Network (SE-Net)
  - [Hu-CVPR-2018, Hu-TPAMI-2019]
  - Chain of SE-blocks
  - Squeeze:
    - Channel descriptor by aggregating over spatial dimension
  - Excitation
    - Small bottleneck fully connected net producing scale of each channel
  - Capture channel interdependences
  - Winner of ILSVRC 2017 (Top-5 err 2.25%)
  - Negligible extra computational cost
CNN architectures

- **MobileNet** [Howard-2017, Google Inc.]
  - depth wise separable convolutions

- **ShuffleNet** [Zhang-CVPR-2018, Face++]
  - Comparable accuracy with AlexNet, 13x speed up
CNN architectures

- Vision Transformers [Dosovitskiy-2021]
  - Taken from Natural Language Processing [Wasvani-2017]
  - No Convolutions
  - Image is cut into fixed-size patches and the sequence of vectorized patches (tokens/words) is fed into the transformer

- Outperforms ResNET on ImageNet, but needs 100M image pretraining
CNN architectures

- (Vision) Transformer
  - Main idea: Self-Attention Mechanism
    - Inputs (vectors $x_1, \ldots, x_m$)
    - Parameters (matrices $W_q, W_K, W_V$)

Query: $q_{:i} = W_Q x_i$,  
Key: $k_{:i} = W_K x_i$,  
Value: $v_{:i} = W_V x_i$.

$c_{:j} = V \cdot \text{Softmax}(K^T q_{:j})$. 

Courtesy of Shusen Wang
CNN architectures

- (Vision) Transformer
  - Main idea: Self-Attention Mechanism
    - Inputs (vectors $x_1, \ldots, x_m$)
    - Parameters (matrices $W_q, W_K, W_V$)

Query: $q_{i:1} = W_Q x_i$, Key: $k_{i:1} = W_K x_i$, Value: $v_{i:1} = W_V x_i$.

Courtesy of Shusen Wang
CNN architectures

- **(Vision) Transformer**
  - Main idea: Self-Attention Mechanism
    - Inputs (vectors $x_1, \ldots, x_m$)
    - Parameters (matrices $W_q, W_K, W_V$)

Weights: $\alpha_{i,j} = \text{Softmax}(K^T q_{i,j}) \in \mathbb{R}^m$. 

\[ \begin{align*}
\alpha_1 & \quad \alpha_2 & \quad \alpha_3 & \quad \ldots & \quad \alpha_m \\
q_1 & \quad k_1 & \quad v_1 & \quad q_2 & \quad k_2 & \quad v_2 & \quad q_3 & \quad k_3 & \quad v_3 & \quad \ldots & \quad q_m & \quad k_m & \quad v_m \\
x_1 & \quad x_2 & \quad x_3 & \quad \ldots & \quad x_m
\end{align*} \] 

Courtesy of Shusen Wang
CNN architectures

- (Vision) Transformer
  - Main idea: Self-Attention Mechanism
    - Inputs (vectors $x_1, \ldots, x_m$)
    - Parameters (matrices $W_q, W_K, W_V$)

Output vectors: \[ c_j = \alpha_{1j}v_{1} + \cdots + \alpha_{mj}v_{m} = V\alpha_j. \]
CNN models (comparison)

CNN models (comparison)

- **ImageNet leaderboard** (Top-1 accuracy)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Model</th>
<th>Top 1 Accuracy</th>
<th>Top 5 Accuracy</th>
<th>Number of params</th>
<th>Extra Training Data</th>
<th>Paper</th>
<th>Code</th>
<th>Result</th>
<th>Year</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CoAtNet-7</td>
<td>90.88%</td>
<td></td>
<td>2440M</td>
<td>✓</td>
<td>CoAtNet: Marrying Convolution and Attention for All Data Sizes</td>
<td></td>
<td></td>
<td>2021</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ViT-G/14</td>
<td>90.45%</td>
<td></td>
<td>1843M</td>
<td>✓</td>
<td>Scaling Vision Transformers</td>
<td></td>
<td></td>
<td>2021</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CoAtNet-6</td>
<td>90.45%</td>
<td></td>
<td>1470M</td>
<td>✓</td>
<td>CoAtNet: Marrying Convolution and Attention for All Data Sizes</td>
<td></td>
<td></td>
<td>2021</td>
<td></td>
</tr>
</tbody>
</table>
Face interpretation problems

- Face recognition, face verification
  - Architecture similar to AlexNet - very deep CNN (softmax at the last layer)
  
  [Taigman-ECVV-2014] DeepFace: Closing the Gap to Human-Level Performance in Face Verification (authors from Facebook)
  
  [Parkhi-BMVC-2015] Deep Face recognition (authors from Oxford Uni)
  - 2.6M images of 2.6k celebrities, trained net available

- Face represented by penultimate layer response, similarity search, large scale indexing
Face interpretation problems

- Facial landmarks, Age / Gender estimation
  - Multitask network
    - Shared representation
    - Combination of both classification and regression problems
Age estimation – How good the network is?

- Our survey
  ~20 human subjects, ~100 images of 2 datasets

**MORPH dataset**
- True: 22, MAE: 18.8
- True: 36, MAE: 17.8
- True: 33, MAE: 16.3
- True: 22, MAE: 16.1
- True: 25, MAE: 16.0

**IMDB dataset**
- True: 25, MAE: 0.5
- True: 66, MAE: 1.0
- True: 29, MAE: 1.0
- True: 19, MAE: 1.0
- True: 43, MAE: 1.0
Age estimation – How good the network is?

- Better than average human...

![Graph showing age estimation performance with MAE, CS5, and MaxAE for different groups: Average human, Best human, Worst human, Human crowd, and Machine.](MORPH and IMDB graphs)

\[
\begin{array}{llll}
\text{Average human} & 6.8 & 48.6 & 24.1 \\
\text{Human crowd} & 4.7 & 65.1 & 19.0 \\
\text{Machine} & 3.2 & 82.6 & 26.0 \\
\text{Average human} & 8.2 & 41.7 & 31.5 \\
\text{Human crowd} & 5.7 & 59.0 & 21.0 \\
\text{Machine} & 5.1 & 62.5 & 42.7 \\
\end{array}
\]

- [Franc-Cech-IVC-2018]
- Network runs real-time on CPU
Predicting Decision Uncertainty from Faces

- Can we train a classifier to detect uncertainty?

=> YES, we can...

- CNN 25% error rate, while human volunteers 45%

Training set: 1,628 sequences
Test set: 90 sequences

![Question and options]

- WHICH OF THE FOLLOWING IS NOT A SUBATOMIC PARTICLE?
  - A: PROTON
  - B: NEUTRON
  - C: BONBON
  - ELECTRON

- SIGH
- CHUCKLES

![Histogram]

Number of correct answers vs. occurrences
Sexual Orientation from Face Images

- Better accuracy than human in (gay vs. heterosexual)
  - 81% accuracy (for men), average human accuracy (61%)
  - 71% accuracy (for women) average human accuracy (54%)
  - Accuracy further improved if 5 images provided (91%, 83%)
General recipe to use deep neural networks

Recipe to use deep neural network to “solve any problem” (G. Hinton 2013)

- Have a deep net
- If you do not have enough labeled data, pre-train it by unlabeled data; otherwise do not bother with pre-initialization
- Use rectified linear units instead of standard neurons (sigmoid)
- Use dropout to regularize it (you can have many more parameters than training data)
- If there is a spatial structure in your data, use convolutional layers

Novel:

- Use Batch Normalization [Ioffe-Szegedy-NIPS-2015]
- ReLU => ELU
- Adaptive Optimizers (ADAM)
- Various architectures (AlexNet, VGG, GoogLeNet, ResNet, ResNeXt, DenseNet, SE-Net, MobileNet, ShuffleNet, Transformers)

Experience:

- Data matters (the more data the better), transfer learning, data augmentation
Conclusions

- CNNs efficiently learns the abstract representation (shared among classes)
- Low computational demands for running, Training needs GPU
- Many “deep” toolboxes: Caffe (Berkeley), MatconvNet (Oxford), TensorFlow (Google), Theano (Montreal), PyTorch (Facebook), …
- NNs are (again) in the “Golden Age” (or witnessing a bubble), as many practical problems seem solvable in near future
- Explosion of interest of DNN in literature, graduates get incredible offers, start-ups appear all the time

- Do we understand enough what is going on?
  http://www.youtube.com/watch?v=LVLoc6FrLi0

Acknowledgement: I borrowed some images from slides and blogs of G. Hinton, A. Ng, Y. Le Cun, Fei-Fei Li, K. He, J. Brownlee, K. Rupp, Shusen W
Further Resources

- Deep Learning Textbook
  - Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016
  - Available on-line for free.

- Lectures / video-lectures
  - Stanford University course on Deep Learning (cs231n)
  - MIT lectures on Introduction in Deep Learning (MIT 6.S191)

- Various blogs and on-line journals
  - Andrej Karpathy (blog)
  - Distill (distill.pub)