Computer vision

introduction and overview of current practice

motivational lecture for the Mozgalo contest

Siniša Šegvić
Faculty of Electrical Engineering and Computing
Sveučilište u Zagrebu
About: Computer Vision

We study techniques to recover useful information from images.

Two main approaches towards that goal are:

- **recognition**: learn to recover symbolic information, eg. is there an object at (x,y)?
- **reconstruction**: estimate numerical information related to geometry, eg. scene structure, camera motion, ...
About: methods

Both approaches heavily rely on advanced optimization:

- **recognition**: optimize a mapping from images to probabilities
  - criterion: performance on the training data (+ regularization...)
  - optimization performed during training, inference (relatively) simple
  - examples: localization, object classification, image classification

- **reconstruction**: optimize a geometrical model
  - criterion: agreement with the observed image
  - optimization is performed during inference, there is no training
  - often require more than one image

Focus of this talk: **recognition**

A little unfair to reconstruction methods, however:

- Mozgalo targets recognition problems
- recognition methods appear to develop faster today
Until recently, image recognition was an academic discipline

- results were reproducible only in carefully controlled environments
- however, the desire to understand machine perception motivated people to carry on

Today, vision by recognition is a technology with clear industrial value

- companies like Google, Facebook, Intel, Microsoft, Nvidia, and many Chinese giants invest billions of dollars each year
- for instance, Google buyed DeepMind for 0.6 B$ (2014)
- for instance, Intel buyed MobileEye for 15.3 B$ (2017)
- NVidia invested 20 M$ in TuSimple (2017)

Let's see what's so hot about vision!
Applications: autonomous cars

Society of automotive engineers defines 6 autonomy levels (2014)

- Tesla autopilot qualifies as level 2 (2016).
- First level-4 vehicles appear: Google, Tesla, Uber, Audi, Renault ...

https://www.youtube.com/watch?v=9ydhDQaLAqM
Applications: automated diagnostics

by showing it thousands of pictures of skin conditions.

APPLICATIONS: AUTOMATED SHOP

https://www.youtube.com/watch?v=Jg0pQaV44I8

Amazon Go, 2131 7th Ave, Seattle, WA.
Applications: automated payments

instant results

https://www.youtube.com/watch?v=oXn9NuUmhDM

https://www.youtube.com/watch?v=XlbVB50mlh4

...and solutions to mathematical problems (with all steps!)
APPLICATIONS: PROJECTIONS AND PREDICTIONS

Computer Vision Revenue by Application Market, World Markets: 2015-2022

- Automotive
- Sports & Entertainment
- Consumer
- Robotics and Machine Vision
- Medical
- Security & Surveillance
- Retail
- Agriculture

Source: Tractica

https://www.tractica.com/
APPLICATIONS: AGENDA

1. Introduction and motivation
   - two basic approaches, applications
2. A look back: classic computer vision
   - on digital images, main milestones
3. The image classification problem
   - approaches, advantages, shortcomings
4. Convolutional models
   - architectures, layers, code
5. Our research
   - overview, datasets, results
6. Hardware support, future trends
7. Conclusion
Digital images: formation

A digital camera consists of three main elements

1. **sensor**: a rectangular array of light sensing elements (→ **pixels**)!

2. **aperture**: a tiny hole which lets the light arrive onto the sensor
   - it ensures that each sensing element gets excited by the corresponding narrow beam of light.

3. **lens**: a light collection device (ensures stronger signal)

For color images, we would need a slightly more complex system
- e.g. three color filters, three sensors and a system of mirrors.
Digital images: pixels

Thus, any digital image is a rectangular matrix of pixels

- dimensions (rows ⨉ columns) depend on the sensor

If we zoom enough into the image, pixels become visible:

(if they don't - instruct your viewer to refrain from smoothing)
In color images, each pixel is an (R, G, B) triplet.

Such image can be represented with a tensor $H \times W \times 3$:

A flattened tensor can be fit into memory as an array of $H \cdot W \cdot 3$ numbers:

An image of $200 \times 200$ pixels can be described with 120000 numbers.
A look back: inception (1966)

Immediately after digital images appeared, people started to wonder:

- can a computer comprehend the world by analyzing images?

Year 1966: an american professor assigns a student project with a goal to develop a program which prints a description of the input image.

The professor was a little in front of the time: the problem had remained unsolved for many years despite diligent work of many people...

Our understanding grew slowly but steadily, main milestones follow.
A look back: object classification (1991)

Approach to classify well-aligned face images [turk91cvpr]:

- treat images as high-dimensional datapoints
- hypothesize that they live on a lower-dimensional manifold
- use training data to learn a projection to the *eigenface* manifold
- perform the recognition task on the manifold representation
A look back: object localization (2001)

Approach to localize objects from a single class [viola01cvpr]:

- treat localization as binary detection in the sliding window
- learn the binary classifier as an attentional cascade
- learn individual levels of the cascade by combining simple features
- could process over 100000 patches at 25 Hz

Approach to localize objects from a single class [dalal05cvpr]:
- linear classification in a sliding window
- represent patches with a powerful local descriptor (HOG)
- more accurate than [viola01cvpr] but slower

Extension of the last two approaches solved the traffic sign localization:

[segvic14mva]

Approach to classify unaligned images [csurka04eccv]:
- code local patch appearance with hand-crafted features (SIFT)
- learn a visual dictionary of patch descriptors (no supervision)
- represent images as histograms over visual words
- train a shallow model to classify such representations

[fergus09iccv]
A look back: yesterday

It became wildly known that vision is hard...

2014: popular culture portrays computer vision as a mission impossible...

□ finding out whether an image contains a bird?
□ well, with a research team and five years - maybe...

However, we came a long way since 1966

□ reliable bird detection was feasible before 2010 (with a research team)
□ methodology and tools of today allow to solve such problems at home!

Classic vision → A look back (5) 19/56
A look back: today

Thus, a month after XKCD 1425 came out, Flickr proudly reported:

- the "park or bird" problem is solved...
- ...in less than five years :-)

https://code.flickr.net/2014/10/20/introducing-flickr-park-or-bird/

Computer science problems are easy to underestimate and overestimate :-)

Classic vision → A look back (6) 20/56
**Classification: problem**

Image classification is the ultimate recognition problem

Hard because we do not know where is the defining object

Especially hard when intra-class variance is large and inter-class variance is small

Consider the problem of discriminating bison from oxen:
We could try to solve image classification by a rule-based system:

1. concentrate on image regions projected from four legged animals
2. oxen have longer horns or no horns at all
3. bison are dark brown, etc, etc

It's neat (no learning!), but nobody succeeded to make that work!
Classification: learning

Hence, we look up approaches which are able to learn functionality from the data.

A machine learning approach would roughly follow the following steps:

1. express the program with many **free parameters**
   - the parameters determine a transformation which we call the **model**
2. fit parameters on the **training set**
3. evaluate performance on the **test set**

Success depends on the model, training set and processing power

1. model may have insufficient (or excessive) capacity
2. the **training set** may be too small or not representative enough
3. insufficient processing power ⇒ the training may not converge
Classification: bag of words

Early image classification approaches (BOW) had three layers:

1. describe patches with hand-crafted feature descriptors
2. learn a visual dictionary (a collection of visual words)
3. map patches into visual words, and aggregate them into a histogram
4. classify image descriptors with a shallow model
5. can be viewed as a kernel approach with explicit embedding
**Classification: Shallow Classification**

Transformation data $\rightarrow$ decision:

1. one linear projection
2. optional squashing non-linearity

An example in 2D: $y_i = \sigma(w^\top x + b)$

Advantages of shallow models:
- the best solution is guaranteed and fast
- this is the best approach when classes are linearly separable

Unfortunately, shallow models are a poor fit for image classification:
- state-of-the-art BOW embeddings are not linearly separable
- insufficient capacity, poor generalization (tendency to overfit)
Classification: data representation

Classification can profit from good representation:

- it would be easy to discriminate bison from oxen if some magical algorithm converted the input image into a binary vector: [fur?, small horn?, wilderness?, ...]
- most bison would be: [1, 1, 1, ...]
- most oxen would be: [0, 0, 0, ...]

Hand crafting quality representation is hard:

- Greek and Romans did not invent 0 in 1000 years of civilization
- that significantly hindered the development of maths
  - MCMLXXI + XXIX = ?
  - MXXIV : LXIV = ?

Best: learn the representation and the classification simultaneously!

[goodfellow16]
**Classification: Compositional Paradigm**

Deep model: a sequence of learned non-linear transformations

Why deep learning was not successful?
- no guarantee of the learning success
- non-competitive performance

Why deep learning became popular?
- better modeling and training
- large datasets ($n=10^6$)
- processing power (TFLOPS)

Useful for understanding images, languages, speech, bioinformatics...

\[
\begin{align*}
\Theta_1 &\rightarrow f_1(x, \Theta_1) \\
\Theta_2 &\rightarrow f_2(h_1, \Theta_2) \\
\Theta_3 &\rightarrow f_3(h_2, \Theta_3) \\
\Theta_4 &\rightarrow f_4(h_3, \Theta_4) \\
p(ox|x) &\quad p(bison|x)
\end{align*}
\]
CONVOLUTIONAL MODELS: ARCHITECTURE

Deep models for image classification typically consist of:

- linear **convolutions** (recognize object parts)
- linear projections (recognize image as a whole)
- poolings (reduce representation dimensionality)
- linear layers followed by an elementwise non-linearity, eg. max(0,x)
CONVOLUTIONAL MODELS: CONVOLUTIONS

Task: convolve the previous feature map with a kernel

We typically have multiple feature maps on output ⇒ multiple kernels
We typically have several feature maps on input ⇒ kernels are 3D
**Convolutional models: poolings**

Task: reduce dimensionality to relax memory requirements

Most often implemented as average pooling or max pooling

It also increases translation invariance:
Deep **convolutional** model for image classification [krizhevsky12nips]

- **input**: image; **output**: distribution over 1000 classes
- **fitness criterion**: average log probability of the correct class
- **structure**: a succession of convolutions and poolings
  - gradual decrease of resolution and increase of the semantic depth
- **recent architectures**: $O(10^2)$ layers, $O(10^6)$ parameters, $O(10^9)$ multiplications for a 224x224 image!
Convolutional models: ImageNet

One of the most popular vision datasets [russakovsky15ijcv]

- annual challenges: classification, localization, detection in video
- we focus on the classification challenge: $10^6$ images, $10^3$ classes
- fine-grained animals, objects, materials, sports, dishes...
- evaluation metric: top-five prediction error (trained human: 5%)
CONVOLUTIONAL MODELS: IMAGENET PERFORMANCE

Recent practice → Convolutional models (5) 33/56
Convolutional models: ImageNet hard examples

Tasks which are difficult for humans [russakovsky15ijcv]:
- fine-grained classification (e.g. 120 breeds of dogs!)
- exotic classes (pulley, spotlight, maypole)

Tasks which are difficult for GoogleNet (2014, 6.7%):
- little and thin objects, filtered and atypical images
- abstraction (a toy hatchet, images with text)
- large intra-class variance, small between-class variance
A deep classification model can be fine-tuned for another (easier) task:

- cut-off the last few layers
- connect the remaining layers with the back end for the new task
- train the resulting model on new images
- inherited layers are well-trained so we can train with less data (few thousands images)
Convolutional models: tasks

- Object localization:
  - detect objects...
  - ...and indicate their location

- Semantic segmentation:
  - label all image pixels...
  - ...with semantic classes

- Stereoscopic reconstruction:
  - label all image pixels...
  - ...with metric distances

- Recognition in video, object tracking, styling and generating images, ...
**Convolutional models: tensorflow**

```python
class TFLogreg:
    def __init__(self, d, m, param_delta=0.5, param_lambda=1e-3):
        self.X = tf.placeholder(tf.float32, [None, d])
        self.Yoh_ = tf.placeholder(tf.float32, [None, m])
        self.W = tf.Variable(tf.zeros([d, m], tf.float32))
        self.b = tf.Variable(tf.zeros([m], tf.float32))
        self.probs = tf.nn.softmax(tf.matmul(self.X, self.W) + self.b)

        self.truelogprobs = tf.reduce_sum(self.Yoh_ * tf.log(self.probs), reduction_indices=1)
        self.cross_entropy = tf.reduce_mean(-self.truelogprobs)
        self.loss = self.cross_entropy
        self.trainer = tf.train.GradientDescentOptimizer(param_delta)
        self.train_step = self.trainer.minimize(self.loss)
        self.sess = tf.Session()

    def train(self, X, Yoh_, param_niter=100):
        self.sess.run(tf.global_variables_initializer())
        for i in range(param_niter):
            loss, _ = self.sess.run([self.loss, self.train_step],
                                    feed_dict={self.X: X, self.Yoh_: Yoh_})
            if i % 10 == 0:
                print(i, loss)

    def eval(self, X):
        probs = self.sess.run(self.probs, feed_dict={self.X: X})
        return probs
```

Recent practice → Convolutional models (9) 37/56
Convolutional models: PyTorch

class LogisticRegression(nn.Module):
    def __init__(self, n_input, n_classes):
        super(LogisticRegression, self).__init__()
        self.W = nn.Parameter(torch.rand((n_input, n_classes)))
        self.b = nn.Parameter(torch.rand((1, n_classes)))

    def forward(self, x):
        return torch.matmul(x, self.W) + self.b

data_x = Variable(torch.from_numpy(np.float32(X)).cuda())
data_y = Variable(torch.from_numpy(np.int64(Y)).cuda())

model = LogisticRegression(D, C)
optimizer = optim.SGD(model.parameters(), lr=learning_rate)
model.cuda()

for e in range(epochs):
    output = model.forward(data_x)

    loss = nn.CrossEntropyLoss().forward(output, data_y)
    loss.backward()

    optimizer.step()
    optimizer.zero_grad()
**Hardware: CPU vs GPU**

CPU core (AVX) can dispatch up to 2 FMA instructions / cycle

- $2 \cdot 2.8$ operations @3GHz $\rightarrow$ 100 GFLOPS
- If a CPU has 10 cores - that's 1 TFLOPS

Modern GPUs achieve 10+ TFLOPS in matrix multiplication

- In practice, the advantage is $\times 50$ due to better memory bandwidth.

Price of training a simple ImageNet model:

<table>
<thead>
<tr>
<th>GPU</th>
<th>TFLOPS</th>
<th>price</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>GTX 1070</td>
<td>6.5</td>
<td>4000 kn</td>
<td>52 h</td>
</tr>
<tr>
<td>GTX 1080 Ti</td>
<td>11.3</td>
<td>7000 kn</td>
<td>31 h</td>
</tr>
<tr>
<td>P100 FP16</td>
<td>25.0</td>
<td>47000 kn</td>
<td>13 h</td>
</tr>
<tr>
<td>DGX-1 FP16</td>
<td>170.0</td>
<td>$129000$</td>
<td>2 h</td>
</tr>
</tbody>
</table>
Best performance/price is obtained on gaming GPUs:

- NVIDIA Titan X: 250W, 12GB, 11 TFLOPS, 3000 GPU cores
- Radeon Vega FE: 300W, 16GB, 13 TFLOPS, 4000 GPU cores
- NVIDIA Titan V: 250W, 16GB, 110 TFLOPS (?), 12nm, 8cm², 21Gt

Actual non-representative measurement:

- GTX1070 (3500 kn) 60× faster than E3-1220 v3 (1700kn)

Additional challenge: deliver such performance with low power
Hardware: Embedded

Novel hardware concept: processing units for artificial intelligence

- fast matrix multiplication (10 TFLOPS)
- low power, low precision (8-32 bit)

Main players:

- NVIDIA TX2: 1.5 TOPS, 15W, 8GB RAM
- NVIDIA Xavier: 20 TOPS, 20W, automotive certificate (ISO 26262)
- Google TPU2: 45 TFLOPS
- Microsoft + Intel DPU (Stratix-10): 40 TFLOPS
Our research: semantic segmentation

Pixel-level image understanding (or semantic segmentation):
- pixel-level: associate each pixel with a class
- image understanding: classes have a high-level meaning

Traffic participants: person (red), car (blue), bicycle (dark red)

Objects: pole (light grey), traffic sign (yellow), traffic light (orange)

Landscape: road (purple), sidewalk (pink), building (dark grey), vegetation (dark green), terrain (light green), sky (light blue)
Our research: return of the sliding window

A classification model can be applied to the segmentation task:

- analyze the image in the **sliding window** fashion
  - each patch produces one pixel of the semantic map
- segmentation groundtruth allows end-to-end training
- each pixel becomes one component of the fitness criterion
- optimized implementation required in practice
  - $10^6$ pixels $\times 10^9$ multiplications?

road: 1%
sidewalk: 2%
...
person: 85%
...
Our research: going fully convolutional

Luckily, the processing of neighbouring patches involves calculating many common latent activations

More efficient: perform the classification layer-wise [long15cvpr]:
- the resulting semantic map is subsampled due to pooling
- this can be relaxed to some extent with dilated filtering [yu16iclr]
Our research: datasets

Cityscapes [cordts16cvpr]
- driver's perspective, 19 classes
- 5000 stereo images, 2MPixel
- 20000 coarsely annotated images
- instance level annotations
- 50 cities, spring to autumn

Vistas [neuhold17iccv]
- driver's perspective, 100 classes
- 25000 images, 2-8 MPixel
- instance level annotations
- worldwide, various weather
Our research: performance characterization

Widely used performance metric: **intersection over union (IoU)**

- set A: groundtruth pixels of class c
- set B: predicted pixels of class c

\[
\text{IoU}_c = \frac{|A \cap B|}{|A \cup B|}
\]

Typically, the performance is expressed as mean IoU over all classes

\[
\text{mIoU} = \frac{\sum_c \text{IoU}_c}{C}
\]

- this increases the influence of rare classes with few training pixels
- examples: wall, fence, pole, bottle, potted plant

To ensure integrity, labels of test subsets are withheld from public datasets

The performance on the test set is determined by submitting results to the evaluation server
Our research: case study

A pedestrian and a cyclist correctly recognized by 6 out of 7 models:
Our research: speed vs accuracy

![Graph showing the comparison of accuracy vs time for different models. The models include ResNet38, PSPNet, DUC, RefineNet, FRRN, Ladder-DenseNet, ICNet, LRR, DeepLabv2, Dilation10, DPN, DeepLab, CRF-RNN, SQ, SegNet, and ENet. The graph plots accuracy on the y-axis and time (frames/second) on the x-axis. The x-axis is log-scaled.]
TRENDS

Embedded applications (TX2, Xavier)

Unsupervised learning and image generation

[radford16iclr]
TRENDS

Future prediction

[luc17iccv]
TRENDS

Image to text translation

A close up of a person brushing his teeth.

A woman laying on a bed in a bedroom.

A black and white cat is sitting on a chair.

[donahue17pami]

Important new work: visual genome dataset
**Conclusion: State of the Art**

Dramatic improvement of prediction accuracy in the last few years:

- **2015 [ros15wacv]**
  - Cityscapes categories: 91% vs 89.7%
  - Cityscapes classes: 81.4% vs 78.4%

- **2017 [kreso17cvrsuad]**
  - Cityscapes categories: 91% vs 89.7%
  - Cityscapes classes: 81.4% vs 78.4%
**Conclusion: outlook**

Performance of computer vision systems will continue to grow

---

**Important research directions:**

- relaxing supervision (unsupervised, semi, weakly)
- estimating prediction uncertainties (adversarial examples)
- learning architectures for visual recognition
- forward pass on low power devices (quantization, distillation)
Conclusion: discussion

Thank you for your attention!

questions?

Our results have been achieved on projects MultiCLoD (HRZZ I-2433-2014) [1] and SafeTram (Končar i ERDF, 2017-2020) [2]:


Ivan Krešo will defend his thesis subject on January 31 at 15h in D306.

☑ please let me know if you plan to come so I can arrange a larger room

We will be hiring a PhD student in July-September 2018

☑ feel free to let me know if you are interested!
CONCLUSION: BIBLIOGRAPHY


CONCLUSION: BIBLIOGRAPHY (2)


