Example 1: ImageNet (visual object classification)

**ImageNet Challenge**

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

**Training set:** \( \mathcal{T}^m = \{(x^i, y^i) \in \mathcal{X} \times \mathcal{Y} | i = 1, \ldots, m\} \), where:
  - \( \mathcal{X} \) are images from ImageNet,
  - \( \mathcal{Y} \) is a set of output classes (ILSVRC 2012 defines \( |\mathcal{Y}| = 1000 \) of them).
Example 1: ImageNet (visual object classification)

**Class of prediction strategies:** VGGNet (Zisserman, et al., 2014), i.e. a convolutional neural network with fixed structure. Note that a convolutional neural network with $p$ layers is a function composition $h(x; \theta) = (f_{\theta_p}^p \circ f_{\theta_{p-1}}^{p-1} \circ \ldots \circ f_{\theta_1}^1)(x)$. Its outputs are interpreted as class probabilities, i.e. $p(y = c \mid x) = h_c(x; \theta)$.

**Loss function:** negative log-likelihood of class probabilities (a.k.a. cross entropy)

$$
\ell(y^i, h(x^i)) = - \sum_{c \in Y} [y^i = c] \log(h_c(x^i)).
$$

**Learning approach:** empirical risk minimisation, gradient descent

$$
R_T^m(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell(y^i, h_\theta(x^i)) \rightarrow \min_\theta
$$
Example 1: ImageNet (visual object classification)

- Results by VGGNet

- More details in lectures on deep learning
Example 2: licence plate recognition

Online app estimating the Travel Time for cars in Prague based on the number plate recognition: https://unicam.camea.cz/Discoverer/TravelTime3/map
Example 2: licence plate recognition

Input image $x \in \mathcal{X}$ of size $[H \times W]$

![ULK 68-39](image)

Model of synthetic license plate images

A set of templates $w = (w_a|a \in \mathcal{A})$ for each character from $\mathcal{A}$

![019ABZ...](image)

A segmentation $y = (s_1, \ldots, s_L) \in \mathcal{Y}(x)$, where $s = (a, k), a \in \mathcal{A}$ is a character code and $k \in \{1, \ldots, W \}$ is a character position, together with templates $w$ defines a synthetic image:

![ULK68-39](image)

An admissible segmentation $y \in \mathcal{Y}(x)$ ensures that the templates are not overlapping and that the synthetic image has the same width as the input image $x$:

$$k(s_1) = 1, \quad W = k(s_L) + \omega(s_L) - 1, \quad \text{and} \quad k(s_i) = k(s_{i-1}) + \omega(s_{i-1}), \quad \forall i > 1$$

where $\omega: \mathcal{A} \to \mathcal{N}$ are widths of the templates.
Example 2: licence plate recognition

- We want a classifier which outputs the segmentations \( y \in \mathcal{Y}(x) \) defining a synthetic image most similar (measured by correlation) to the input image \( x \):

\[
\hat{y} = h(x; w) = \arg \max_{(s_1, \ldots, s_L) \in \mathcal{Y}(x)} \sum_{i=1}^{L(y)} \sum_{j=1}^{\omega(a(s_i))} \langle \text{col}(x, j + k(s_i) - 1), \text{col}(w_{a(s_i)}, j) \rangle
\]

- **Problem:** How to construct the templates \( w = \{w_a | a \in A\} \) so that the classifier \( h(x; w) \) predicts a segmentation with small Hamming distance to the correct one?

- **Solution:** Select the templates \( w \) so that the classifier \( h(x; w) \) performs well on a training set \( \{(x^1, y^1), \ldots, (x^m, y^m)\} \) and simultaneously control the over-fitting.

- More details in the lecture on **Structured Output Support Vector Machines**.
Example 3: Joint segmentation & registration

**Given:** set of images, each containing an object instance, and a shape model

![Images of various objects](image1.png) ![Images of various objects](image2.png) ![Images of various objects](image3.png)

**Task:** segment & register each image to the reference frame (shape model)

- image $x = \{x_i \in \mathbb{R}^3 \mid i \in D'\}$, binary segmentation $y = \{y_i \in \{0, 1\} \mid i \in D\}$
- shape model $p(y) = \prod_{i \in D} p_i(y_i)$, with Bernoulli distributions $p_i(y_i = 0, 1)$.
- appearance model $p_\theta(x_j \mid (Ty)_j), j \in D'$, where
  - $T$ is an affine transformation,
  - $p_{\theta_0}(x_j \mid y'_j = 0), p_{\theta_1}(x_j \mid y'_j = 1)$ are two mixtures of Gaussians.
Example 3: Joint segmentation & registration

- loss function $\ell(y, y') = \sum_{i \in D} \mathbb{I}\{y_i \neq y'_i\}$, i.e. Hamming distance

(1) Segmentation for **known** $T$ and $\theta$: minimise expected Hamming distance between true and estimated segmentation $\Rightarrow$

$$y = h_{T, \theta}(x) = \{h_i(x) \mid i \in D\}$$

$$h_i(x) = \arg\max_{y_i=0,1} p_\theta((T^{-1}x)_i \mid y_i) \cdot p_i(y_i)$$

(2) How to estimate unknown $T$ and $\theta$? See lecture on the **EM-Algorithm**.