Zkouška

Video formats

Digitalization

- Analog → AD converter → Digital → DA converter → Analog
- Audio - array of numbers
- Video - 3d array of rgb tuples
- **Sampling** - with an appropriate frequency, quantized based on required bit depth
  - too low loses detail, too high generates more data - quality vs storage
  - **aliasing** - losing information when sampling, creating unwanted signals
  - At least 2 times the frequency of the original signal
    - humans can hear 20-22 kHz → 44 kHz for sound sampling
- **Bit rate** (bps) - number of bits transmitted/processed per second
  - audio - higher bitrate = better quality as more data can be used to represent sound, but also larger size
  - video - higher bitrate = better visual quality, larger file size and streaming bandwidth requirements

Containers

Multi-media object

- Sequence of images + different resolutions
- Audio stream + different languages
- Subtitles
- Meta-data
- Detected object
- Cover image

Container

- Single standardized file comprising all related data
  - synchronization info, fragmentation, data structure, etc.
- Compression is necessary for video and audio
  - **Bit rate** = number of bits to represent timed data per time unit

How to play an encoded file

- Media player transforms a media container into a data stream using a CoDec
- **CoDec** - Coder+Decoder - SW or HW tool for video coding based on some standardized format

MP4 container

- A series of boxes/atoms which contain data
  - Each atom looks like this: `[size][name][data]`
- `ftyp` - FileTypeBox - brand, version, compatibility
- `mdat` - timestamped audio/video payload
- `moov` - metadata (creation time, duration, track types)
  - `stbl` - **sample table boxes**: codec info, sample location in mdat
    - `stsd` - description, codec info
    - `stts` - decoding timestamps and sample durations
      - sample number ↔ decoding timestamp, allowing for playback synchronization
    - `stsc` - sample to chunk
      - how samples are grouped into chunks

        - `stco` - chunk offsets
            - byte offsets of each chunk within the media track
        - `stsz` - sample sizes
            - size of each sample within the media track, determines the amount of data to be read for each sample during playback or processing
- **Chunk** - contiguous block of data, e.g. a video frame, a bunch of audio samples
- **Sample** - snakkest ybut if tge nedua
- You need a CoDec to extract compressed a/v content

Encoding

CoDec

- codec implementations must comply with specs
- decoder implementations can read data from different coders
- best quality for a given bit rate, decoding/encoding time (e.g. for live streaming)
- Original data - list of BMP images based on fps
    - `decoder(coder(frame))=frame` - time/quality trade-off
- Popular codecs
    - 262 MPEG-2 - DVD
    - 264 AVC - Blue-ray, youtube
    - 265 HEVC - 8k video
    - 266 VVC - better HEVC
- Better codecs give higher subjective quality for lower bitrates

Why codecs work?

- human eyes don't percieve the difference, e.g. we don't percieve high-frequency data
- spatial/temporal redundancy - when color stays the same between the frames
- statistical redundancy - we can use less bits for storing common colors

HEVC coding

- Hierarchical structure of dividing images into smaller blocks, allowing for efficient compression
- **Coding tree unit** CTU (HEVC)/**Macroblock** (AVC)
    - up to 64×64px, allow compression of large areas of uniform color or texture
    - each CTU square is divided into 4 equal squares (**quad-tree division**) until the smallest allowable block size (**Coding Unit**) is reached
- Coding tree units → Coding Units (CU), Prediction Units (PU) and Transform Units (TU)
    - Coding unit size varies from 8×8 to 64×64, adapts to different levels of detail within a frame
    - **Prediction units** define how prediction is performed within CU
        - 2Nx2N - same size as CU, no further partitioning
        - NxN - CU is split into 4 equal parts, each forming a PU
        - NxnU, NxnD, nLxN, nRxN - asymmetric motion partitions, more flexible, non-uniform motion
    - **Transform units** - transformation and quantization of residual data after prediction
        - 4×4 to 32×32px, capture frequencies of the remaining data
- Summary:
    - CTU division - 64×64px
    - CU partitioning - subdivision based on texture, detail and motion
    - PU partitioning - subdivision of CUs based on spatial similarity
    - TU partitioning - subdivision based on frequencies
- **Intra-frame** (spatial prediction) - within same frame, jpeg-like,
    - RGB → YCbCr → Chroma subsampling → Frequency-Domain → Quantization → Entropy encoding

        - YCbCr - Y-luma, Chroma-blue, Chroma-red
        - **Chroma subsampling** - compress Chroma and keep Luma because our eyes are more sensitive to brightness than color
        - **Frequency domain** - our eyes are less sensitive to high-frequency details, Discrete Cosine Transform to extract separate frequencies
            - **Energy compaction** - largest DCT coefficient values are concentrated in low-freqency coefficients
            - More coefficients = better quality
        - **Quantization** - divide and round values using a quantization table
            - defines compression quality - lower=worse
        - **Huffman entropy encoding** - our data has lots of zeroes which can be compressed as well
    - Prediction
        - Vertical, horizontal, angular, DC, planar prediction - predicting each pixel based on the one before it
        - 👍 Works well when we have uniform blocks of pixels
        - 👎 but not so much when we have high noise or fine details
- **Inter-frame** - between different frames
    - **Motion estimation** - encoder detects movement and searches for motion vectors
    - **Motion compensation** - predicted frame is created by shifting blocks according to motion vectors
    - **Residual calculation** - difference between two frames, allows for better compression (e.g. subtracting same frames gives you a black rectangle)
    - `c(motion info) + c(residual) < c(original)`
    - Sub-pixel precision for better results
    - Tricks
        - Use heuristics to search for reference blocks
        - Use more reference frames to estimate motion
        - Filter out compression artifacts

Audio CoDecs

- input: array of numbers samplet with a given frequency
- `decode(encode(audio_sample)) = audio_sample`? - size/quality trade-off
- MP3 (lossy perceptual-based compression)
- AAC (better for lower bit rates)

Similarity search

- Video - all frames
- Shots - groups of similar frames
- Representative frames - represent shots

Similarity model queries meta-data ↔ content

- attributes
- keywords
- free-form text
- sketch
- example image

Similarity search model

- $$\delta(f_q(query), f_d(data))$$

- distance function between two math objects (e.g. vectors, tensors, sets)
  - $\delta(x, y) \geq 0$
  - $\delta(x, y) = \delta(y, x)$
  - $\delta(x, x) = 0$
- Common functions
  - **Cosine** $1 - \frac{xy}{||x||\,||y||}$
  - **Lebesgue** $\sqrt[p]{\sum_{i=1}^{n} w_i |x_i - y_i|^p}$
    - L1=Manhattan
    - L2=Euclidean
    - L_∞=maximal
  - **Jaccard** - set similarity, intersection minus union
    - $1 - \frac{|X \cap Y|}{|X \cup Y|}$
- Similarity queries
  - **Range** - all objects within a given radius
  - **kNN** - nearest k objects to a query

Computing similarity scores

- **Global representation** - e.g. a vector representing entire image
  - 👍 efficient, simple, scalable
  - 👎 loses detail, less robust to small variations
  - Histogram
    - Bins: hard - one bin per step / soft - more bins per step
    - Quantization - q-levels, image resize
    - Channel combinations (joint histogram)
    - Location - different image parts
    - For a video sequence - more frames into one histogram
- **Local representation** - e.g. image keypoints and region descriptors, SIFT, SURF
  - 👍 precise, flexible, fine-grained
  - 👎 complex, less scalable

Convolutional neural network (CNN)

- **Convolutional layers** - applies filters (kernels) by sliding over the input image and computing dot products between the filter and the local regions in order to detect edges, textures and shapes
  - **Filter** - small matrix (e.g. 3×3, 5×5) of weights designed to detect specific patterns such as edges, textures, or more complex as the layers deepen
    - $$f_\Theta : R^{w \times h \times 3} \rightarrow R^{|C|}$$
      - maps image to the list of C classes
    - Each filter generates a channel
    1. Extract a $k \times k$ patch from the image
    2. Multiply each element of the patch by the corresponding element of the filter K
    3. Sum the products to get a single value and place it into the output matrix O
  - **Convolution** - process of sliding the filter over the input image, performing multiplication and summing the results to produce **feature maps**
  - Filter values can learn and adjust through the process called backpropagation
- **Activation function** - is applied after convolution, introduces non-linearity
  - $ReLU(x) = max(0, x)$ - outputs positive inputs directly, otherwise 0, applied element-wise

    - Linear functions can only represent proportional change and cannot capture complex patterns, e.g. you are limited by a straight line in classification while non-linear functions enable you to create complex curves to separate the data points
- **Pooling layers** - reduce spatial dimensions of the previous feature maps for efficient computation, max/average pooling
    - outputs resized "image" (actually its feature map)
    - **max pooling** - slides a window over input feature map, outputs max value within that window for each position
- **Fully connected layers** (Dense layers) - aggregate features learned by convolutional layers and classify
    - **Softmax** - activation function that transforms scores into a probability distribution over different classes, often used in the last layer
    - ⚠ We don't always use FC layers because they require much more parameters and are computation-intensive, convolutional layers tend to be much more efficient
- **Embedding** - transformation of input data into a more manageable vector space
    - Can be extracted from data, e.g. extracting image features via ResNet for classification
    - Embeddings can represent different objects detected in an image or be used for computing distances in similarity search
    - Embedding layer maps input indices to vectors - can be trained from scratch or initialized with pre-trained embeddings
- More layers = ability to detect more complex shapes from the building blocks in the previous layers

AlexNET

- Image classification, 1000 categories in the ImageNet dataset
- Architecture - 8 layers
    - Input - image
    - 1-5 Convolutional layers
    - 6-8 Fully connected layers

Local image descriptors

- We want object detection to be robust to occlusions, lightning conditions, scaling, rotation, position, etc.
- **Scale-invariant feature transform (SIFT)** - popular pre-DL unsupervised method
- Image is represented by a set of local descriptors (e.g. vectors)
    - v_i represents content of a selected subregion around detected keypoint at (x,y)
    - pairing between two sets of descriptors based on distance $d(v_i, v_j)$, expensive verification in 2D
- Operates on multiple image scales to detect keypoints invariant to scale changes
    - grid of pixel blocks around a keypoint
- DoG (difference of gaussian) pyramid is used to **identify scale-space keypoints** by comparing adjacent scales
    - For each keypoint we take the image put through a gaussian filter (blur) at corresponding scale p and compute intensity differences between scales, keeping the points which are over our threshold
- Once keypoints are detected, SIFT does **descriptor generation** which captures features invariant to scale/rotation/illumination changes
    - Compute **dominant orientation** of each keypoint by taking the highest point in the histogram of orientations, (10, 20, 30, ..., 360) degrees (orientations closer to the center contribute more)
    - Each **descriptor** has location, scale, orientation and a feature vector

    - Compute orientations relative to the dominant orientation, 45-degree differences, 8 histogram bins
    - The 8-bin keypoint histograms are then concatenated into a high-dimensional (128) descriptor that represents keypoint's local appearance
- Object detection
    - represent one image with one global histogram
    - select candidate db images using some efficient index for the global histograms
    - **re-ranking with optimal transformation**
        - For the query image and candidate images, find the optimal transformation by solving an equation
        - Transformation that maximizes the number of **inliers** ($|x' - u|^2 + |y' - v|^2 < \alpha$, points that agree with the transformation) is considered optimal
- Feature vector - array of numerical values that represent features of given data
- Descriptor - representation that summarizes important features, often in computer vision, e.g. edges, corners, textures, shapes - synonymous to feature vectors in the context of SIFT, designed to be invariant to transformations
- Other sources:
    - https://www.baeldung.com/cs/scale-invariant-feature-transform

Shot detection

- traditional approaches
    - similarity of two consecutive frames
    - keypoint tracking
    - ML using hand-crafted features
- Deep learning
    - 3D CNNs
- TransNet - a neural network for fast detection of common shot transitions
- TP = true positive, FP= false positive, FN= false negative, P = precision, R = recall
    - $P = TP/(TP + FP)$
    - $R = TP/(TP + FN)$
    - $F_1 = 2PR/(P + R)$
- **Precision** - accuracy of positive predictions
    - High precision ⇒ low false positives
- **Recall** - sensitivity, ability to capture all positive instances in the dataset
    - High recall ⇒ low false negatives
- **F1 score** - harmonic mean of precision and recall

ResNet

- Residual network architecture - allows for even deeper networks with improved performance
- Instead of learning direct input-output mappings, it learns **residual mappings** - the difference between the desired output and input to a given layer
- **Residual block** - consists of 2+ convolutional layers with shortcut connections (**identity mappings**) that bypass one or more layers, residual blocks are deeply stacked

Object detection with **R-CNN** (region-based cnn)

- localization and bounding box estimation
    1. Generate region proposals (candidate boxes) using selective search
    2. Feed each region through a pretrained CNN to extract features
    3. Classify regions and generate a bounding boxes
- **Selective search** - region proposal algorithm, generates candidate bounding boxes from an image

    - Segments image into a large number of small, homogenous regions based on various features, e.g. color, texture, instensity
    - Iteratively merges similar regions using hierarchical clustering to generate larger regions
    - Merging is guided by a similarity measure ensuring that similar adjacent regions are merged
- **EdgeBoxes** - generates object boundaries based on edge density
- **YOLO** (you only look once) - divide image into a grid and predict bounding boxes and class probabilities directly from grid cells
    - predicts for objects whose centers fall within the cell
- **Faster R-CNN** - introduces RPN, **region proposal network**
    - RPN shares convolutional features with object detection stages
    - RPN generates candidates directly from extracted feature maps
    - slides a small network over the feature map and predicts scores and box offsets
    - The Image first passes through the backbone network to get an output feature map, and the ground truth bounding boxes of the image get *projected* onto the feature map

Popular datasets

- ImageNet - image classification, single object localization, object detection
    - images for concepts in WordNet hierarchy
- COCO - object detection and segmentation

Multi-modal search

Modality

- Specific type of data, e.g. image, text, audio
- (δ_M, f_{e_M}) - **distance metric** and the **embedding function**
    - embedding function maps items from their original form onto feature vectors
    - e.g. a deep neural network can map images onto visual feature vectors, for text Word2Vec can convert text into numerical vectors
- Video data has many modalities - colors, edges, trajectories, sounds, text metadata

Fusion

- a way of combining data of different modalities
- **Early fusion** (feature-level fusion) - combine features from multiple modalities into a single features, e.g. concatenating feature vectors
    - More effective for interdependent modalities, allows models to learn the combinations
    - E.g. whether a social media post is positive or negative based on text and pictures
- **Late fusion** (decision-level fusion) - combine prediction outputs of separate models, e.g. by averaging predictions
    - More effective for distinct modalities or when they have separate processing
    - E.g. multimedia content search, can search by text or image or both, flexible, modular

Mono vs Multi-modal similarity search engine

- **Mono-modal**
    - Input: single modality descriptor (e.g. shape, color or text)
    - Data preprocessing - feature extraction
    - Similarity search - using distance function on indexed data
    - Retrieving IDs - of the most similar objects, then fetching full images and showing the results
- **Multi-modal**
    - Input: combination of multiple modality descriptors
    - Data preprocessing - features from different modalities are extracted and combined into a multi-modal indexed data structure (in early fusion models)

- Similarity esarch - combined descriptors use combined distance functions (e.g. combining distances for shape and color)

Temporal queries

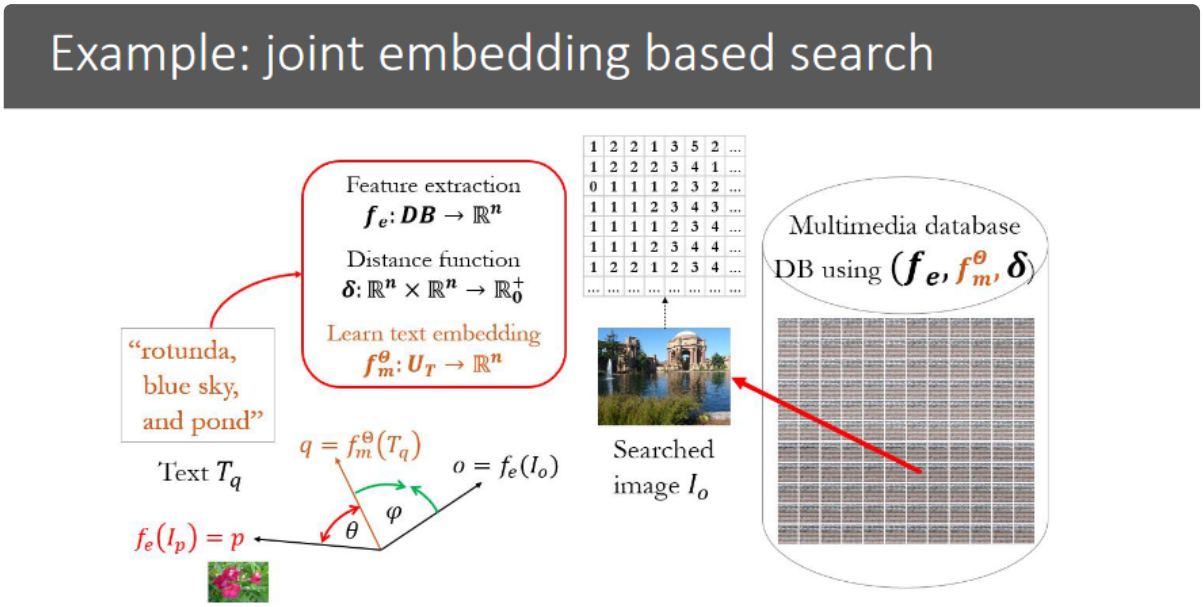
- Model can interpret and respond to queries that involve sequences of events or actions
  - e.g. Person entering a room and sitting down
- Search for a video segment where "sea" frames are followed by "building" frames
- **Temporal context size** $(i + c)$ - i=frame index, c=context size
  - determines number of frames or timeframe units after the initial query ("sea"), the system continues to search for the next item in the query ("building"), can be adjusted

Basic Classification-based Search

- Creates confidence scores for supported classes
- Text query consists of class names, engine rankes images using confidence scores

Joint embedding

- Projecting both modalities onto a single feature space
- $\delta(f_{e_q}(\text{query}), f_{e_d}(\text{data}))$
  - f_{e_q}: input query → feature vector
  - f_{e_q}: data item → feature vector
  - δ : how close are query and data in the same feature space
- Example:
  - "a red sports car" text → feature vector
  - car images → feature vectors
  - text vector is then compared to image vectors in the database



-
  - DB contains preprocessed image vectors, during search a query feature vector is computed and compared
  - T_q - text query to be translated into a feature vector
  - $f_e : DB \rightarrow \mathbb{R}^n$ - extracts image vectors
  - $f_m^\Theta : U_T \rightarrow \mathbb{R}^n$ - extracts text vectors into the same feature space as image vectors
  - $\delta : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_0^+$ - measures similarity between feature vectors
- Loss functions for training joint embedding models
  - Usually the models of different modalities are trained together to ensure proper feature vector alignment

- **Loss function** - measures how well predictions match true data labels, quantifies error between predicted and actual values
- **Contrastive loss** - considers pairs of objects from the same class and pairs of objects from different classes
  - Minimize: $\delta(x, y) + \cdots + max(0, c - \delta(z, w))$
    - x,y - similar
    - z,w - dissimilar
    - c - margin, minimum distance between positive pairs and maximum distance between negative pairs
  - Consider similar and dissimilar pairs of data points to bring the similar pairs together and push the different pairs apart
  - Minimize distancebetween similar, maximize between opposite, some margin is enforced
- **Triplet loss** - considers triplets where one object is from the same class (anchor) and one from a different class
  - Minimize: $max(0, c + \delta(x, y) - \delta(x, z)) + \ldots$
    - x,y - similar
    - x,z - different
    - c - margin
  - Reduce distance between anchor and positive point while increasing the distance between the anchor and the negative point by at least a margin

Contrastive Language-Image Pre-Training (CLIP)

- CLIP jointly trains image and text encoders to predict the correct pairings of batch examples
- **Contrastive learning** - maps similar images and text close together while pushing dissimilar pairs apart
  - Purpose: align text and image representations
  - Use a large dataset of image-text pairs
  - Image encoder & Text encoder
  - Maximize similarity between correct pairs, minimize between mismatched
- **Dataset classifier from label text** creation from label text
  - learns to classify images based on unseen text
  - input image ↔ text labels similarities
  - understand and categorize images according to labels it has never seen during pre-training
- **Zero-shot prediction**
  - model makes predictions on data it has not been trained on
  - recognizes images it hasn't seen yet based on textual descriptions
  - by generating embeddings for both text and images, it can compare them to classify images into categories that were not part of initial training, by simply using the descriptive text of what the image contains
- Robustness
  - bidirectional understanding of text and images - CLIP can associate images with their textual descriptions and vice versa
  - transferable representations - CLIP learns transferable representations during pre-training which can be generalized, they capture high-level semantics
  - large pre-training dataset
  - multimodal contrastive training - enables CLIP to associate similar images with text descriptions
  - zero-shot - CLIP is strong even for datasets it hasn't been trained on
- 👍 versatile - no task-specific data needed
- 👍 scalable, powerful - trained on a diverse range of data
- 👍 efficient - reduces need for extensive dataset-specific training by generalizing the model

Interactive search

Motivation

- **Interaction** gives the search model feedback
- **Exploratory browsing** - improved effectiveness through dynamic interaction
- **Visualization** can help better understand the data
- System interface must be **intuitive and responsive** (indexing!) but it's hard to evaluate subjective experience
  - **qualitative** - user satisfaction, usability studies
  - **quantitative** - speed, accuracy, relevance

Relevance feedback loop

- Feedback
  - Positive/negative
  - Obtained from the current results based on user actions
- **reinforcement learning** - system's effectiveness improves over time
- Use of feedback
  - Update **query vector** based on +/- examples (e.g. Rocchio)
    - moving query closer to preferred items, away from less relevant
  - Update **decision model** (decision hyperplane)
    - modifying the underlying decision-making processs, e.g. ml model parameters
  - Update **relevance scores of all items** (Bayes)
    - statistical methods to update relevance scores of db items

Rocchio algorithm (1971)

- Data represented by vectors
- relevant D_r, irrelevant D_n
- $$query_{new} = (\alpha \cdot query_{old}) + (\frac{\beta}{|D_R|} \cdot \sum_{v_j \in D_R} v_j) - (\frac{\gamma}{|D_n|} \cdot \sum_{v_j \in D_n} v_j)$$
- α, β, γ - importance coefficients, often chosen empirically
- $\beta \cdot \frac{\sum_{v_j \in D_R} v_j}{|D_R|}$ - average vector of relevant documents scaled by β
- Positive reinforcement in the second coefficient pulls the query vector closer to the centroid of the relevant documents
- Negative reinforcement pushes query away from the centroid of the non-relevant documents
- The reused query is more likely to retrieve items similar to those marked as relevant

Active learning

- constant training
- support vector machines to train decision plane

Bayesian feedback loop

- Display D_t in iteration t - set of images based on search query and prior interactions
  - User interacts by picking an image
  - Iterate over all items in the database and update score
  - $\forall I_{o_i} \in DB : \sigma_{t+1}(I_{o_i}) = P(I_{o_i} = I_{os} | H_t)$
    - $H_t = \bigcap_{i=0}^{t} \{D_i, I_{q_i}\}$ - search history of user transactions
  - H_t is updated to include the current iteration
  - select $D_{t+1} \subseteq DB$ - new set of images to display
  - Repeat

- For each object in the database
  - $$P(I_{o_i} = I_{o_S} | I_{q_t}, D_t, H_{t-1}) = \frac{P(I_{q_t} | D_t, H_{t-1}, I_{o_i} = I_{o_S}) \cdot P(I_{o_i} = I_{o_S} | H_{t-1})}{P(I_{q_t} | D_t, H_{t-1})}$$
  - I_{q_t} - image user picked
  - $P(I_{q_t} | D_t, I_{o_i} = I_{o_S}) \sim \frac{\text{similarity}(I_{o_i}, I_{q_t})}{\sum_{\forall I_{d_i} \in D_t} \text{similarity}(I_{o_i} = I_{d_i})}$ - **User model**, notice it in the upper part of our Bayes fraction but without history
    - below is a normalization coefficient, accounts for influence of other images in the display set on the perception of similarity to query - if all images are similar to the query it decreases the probability and we update a bit less
    - $\text{similarity}(I_{o_i}, I_{o_j}) \sim e^{\frac{||f_e(I_{o_i}), f_e(I_{o_j})||^2}{\sigma^2}}$
  - Other notes:
    - P that it is picked in the previous step (we know it)
      - 1/|DB| by default
    - We don't know the user model
      - no H_{t-1} - we expect independence
    - P of choosing I_{q_t} considering display D_t and this is what we want ($I_{o_i} = I_{o_S}$)
    - σ - $\sigma_0 = 1/|DB|$
    - target - o_s
- Model adjusts the percieved relevance of images based on both direct similarity to the query and the influence of other images in the search context - integrates prior knowledge over time

Display selection strategies

- effective selection strategies can help reduce user's cognitive load
- **Most probable items** - items most likely to be relevant based on scores
  - 👍 allows users to find the most relevant images quickly
  - 👎 flawed because user can get stuck in a cluster
- **Score-based sampling** - select items based on relevance probability
  - 👍 more diverse, allows for more exploration, does not always show top-scored items
  - 👎 introduces randomness
- **Entropy-based criterion** - based on set of sampled displays, more balanced
  - choosing pictures most far away from each other
  - maximizes information gain from user feedback by selecting images based on uncertainty of model's predictions
  - 👍 improved performance and maximal information gain for faster improvement
  - 👎 sometimes displays less relevant images, more ambiguous images

Visualization of multimedia data

Limits of human perception

- Limit to how many images can a human perceive in one page
- Response times should be adequate to prevent lagging

Different visualizations for different tasks

- **Finding** items - found should be highlighted
- Content **understanding** - represent overarching themes and patterns that give insights into the entire dataset
- Exploratory **browsing** - no specific target, easy and intuitive discovery and browsing

Gestalt laws

- how humans perceive visual elements as organized patterns rather than separate components
- Law of **simplicity** - complex is perceived as simplest
- Law of **similarity** - similar items are grouped together
- Law of **proximity** - nearby items are grouped together
- Law of **continuity** - eye is drawn to continuous lines
- Law of **closure** - mind seeks to fill in the gaps to create whole
- Law of **symmetry** - symmetrically arranged objects are perceived as forming around some center point

Grid-based presentation

- Standard grid
- Emphasized top corner items
- Progressive disclosure - zigzag pattern
- Temporal/contextual - sequential view, relation across states/times
- Case study on gaze patterns - most users look at the screen center

Reorganizing for larger displays

- When there are too many images, it may be beneficial to do some additional reorganization
- Swap images if it improves a cost function

$$argmax_L \sum_{\forall s,t \in \Omega} \frac{(||P(L_s) - P(L_t)|| - \overline{P})(\delta(s,t) - \overline{\delta})}{\sigma_p \sigma_\delta}$$

  - L - layout
  - $\forall p, s$ - sum over all data points within the dataset Ω
  - $P(x)$ - location of cell x
  - $\overline{P}$ - mean Euclidean distance between any two assigned cells
  - $||P(L_s) - P(L_t)||$ - Euclidean distance between two specific points
  - $\delta(s,t)$ - dissimilarity between two specific points
  - $\overline{\delta}$ - mean dissimilarity between any two data entries
  - σ_p, σ_s - corresponding standard deviation values
  - Numerator: High if both deviations are either positive or both negative, indicating a positive correlation
  - Denominator: Normalizes product by std

Self-sorting map

- an algorithm that organizes data points on a grid so similar data points are located near each other, creating a map
- Reflects patterns and similarities without predefined categories
- Typically generated via unsupervised learning
- Applications
  - pattern recognition
  - feature mapping
  - anomaly detection
- $$T_i = \sum_{k=-1}^{+1} Gau(k) \frac{\sum_{S \in B_{i+k}} S}{|B_{i+k}|} argmin_{(s,t)}(||s - T_i|| + ||t - T_{i+1}||)$$
  - $k \in [-1, 0, 1]$ - we consider neighboring bins, B_{i-1}, B_i, B_{i+1}
  - $Gau(k)$ - Gaussian weighting function at k - higher importance to bins closer to i

    - $\frac{\sum_{S \in B_{i+k}} S}{|B_{i+k}|}$ - centroid, average of the points S in the bin B_{i+k}
    - $argmin \dots$ - finds pair of points s, t that minimizes combined distance from s to the new position T_i and from t to position T_{i+1} to align with the surrounding structure
- The resulting map weighted by Gau is smooth and continuous
- 2D variant:

$$T_{i,j} = argmin_{t \in B_{i,j}} \left[\sum_{k,h=-1}^{1} \left(Gau(k,h) \sum_{s \in B_{i+k,j+h}} \delta(s,t) \right) \right]$$

- To avoid getting stuck in a local minimum, we enumerate all 4!=24 possible alignments of quadruples and find one that minimizes $argmin_{(s,t,u,v)}(\delta(s, T_{i,j}) + \delta(t, T_{i+1,j}) + \delta(u, T_{i,j+1}) + \delta(v, T_{i+1,j+1}))$

Self-organizing map (SOM)

- Artificial NN trained using competitive learning, 2D grid of neurons is fitted to data
- Key concepts
    - **Neurons** - 2d grid, each has a weight vector of the same dimension as input data (position)
    - **Weight vectors** - represent positions in space, adjasted to training data
    - **Topology** - grid structure of the map, how neurons are connected and how they influence each other
    - **Best Matching Unit** - for each data point, the neuron with the weight vector closest to the input vector is BMU
- Training algorithm
    - Init weights
    - Input vector selection - randomly from the dataset
    - Finding the BMU - compute distance between input vector and all weight vectors
        - identify the neuron (BMU) whose weight vector is closest to the input vector by some distance function
    - Updating weights - adjust positions of BMU and neighboring neurons to make them more similar to input
    - $W_i(t+1) = W_i(t) + lr(t) \cdot h_{BMU,i}(t) \cdot (V(t) - W_i(t))$
        - $W_i(t)$ - weight vector of neuron i at time t
        - $lr(t)$ - learning rate at time t
            - linearly decrease lr for network stabilization
        - $h_{BMU,i}(t)$ - neighborhood function, decreases with distance from the BMU over time
            - $BMU_i = argmin_i L_2(V, W_i)$ - finds neuron with weight (position) closest to the input vector
            - $e \frac{-L2(coord(BMU_i), coord(i))}{\sigma^2}$
        - $V(t)$ - input vector at time t
- Applications:
    - Data visualization
    - Clustering, dimension reduction

Approaches to visualization of dataset properties

- for structure preservation while mapping high-dimensional structures into lower dimensions

1. High-dim feature space with manifold structure
2. Mapping without structure preservation in visualization space (could mislead)
3. Mapping with structure preservation in visualization space (colors represent ordered/sequential relationship)

General dataset visualization properties

- **Overview** requirement - visualization should give a faithful overview of the distribution of images in the collection
- **Structure** preservation requirement - relations between images should be preserved when visualizing data
- **Visibility** requirement - all displayed images should be visible to extent that the user can understand the image content