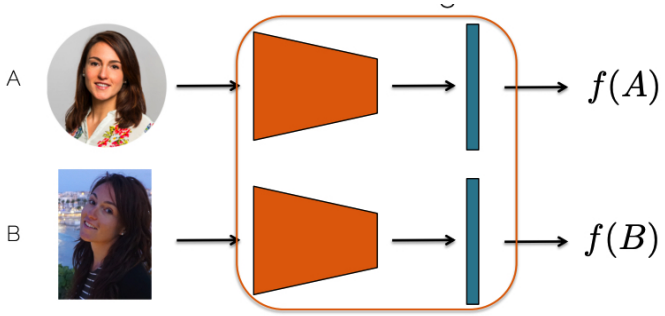
Advanced Deep Learning for Computer Vision

1 Recap: Introduction to Deep Learning

1.1 Siamese Network

- Idea: Train a network to learn a similarity function between two inputs
- Architecture: Two identical subnetworks sharing weights, process the two inputs separately
- Output: A distance metric (e.g., Euclidean distance) between the two feature vectors produced by the subnetworks
- Loss Function: Contrastive Loss or Triplet Loss to encourage similar inputs to have smaller distances and dissimilar inputs to have larger distances
- Applications: Face verification, signature verification

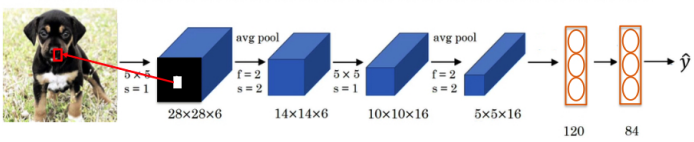
Distance function: $d(A, B) = \|f(A) - f(B)\|_2^2$

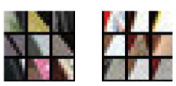


1.2 Visualization of ConvNets

Visualization in the Image Space

- Layer 1: basic geometric shapes
- Deeper Layers: more complex patterns and object parts
- Deeper layer have a much higher resolution since they display the receptive field



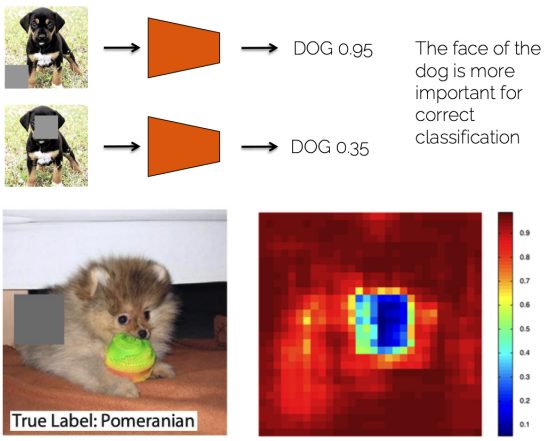


Feature map 2, layer 1, 9 image patches that provided the highest activation

Zeiler and Fergus. „Visualizing and understanding convolutional neural networks". ECCV 2014

Visualize Importance: Occlusion Experiment

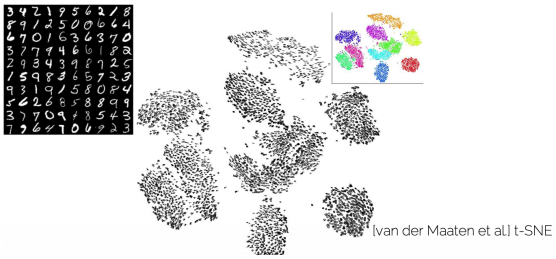
- block different parts in the image and see how classification score changes
- heatmap shows which parts of the image are most important for classification (lower score = more important)



1.3 Visualization of Clusters

t-SNE: t-Distributed Stochastic Neighbor Embedding

- Non-linear dimensionality reduction technique
- Maps high-dimensional data to a lower-dimensional space (typically 2D or 3D) while preserving pairwise distance of local points
- Perplexity parameter controls the balance between local and global aspects of the data
- Not useful for new data points (non-parametric)



1.4 Autoencoders and VAE

Autoencoders

- Unsupervised approach for learning lower-dimensional feature representations
- Consist of an encoder (maps input to latent space) and a decoder (reconstructs input from latent space)
- Trained to minimize reconstruction error (e.g., Mean Squared Error)
- Can be used for dimensionality reduction, de-noising, and generative modeling
- Bottleneck layer: dim (latent space) « dim(input space)

Variational Autoencoders (VAE)

- Probabilistic extension of autoencoders
- Encoder outputs parameters of a probability distribution (mean and variance) instead of a single point in latent space
- Latent space is continuous, allowing for smooth interpolation and generation of new samples
- Loss function includes reconstruction error and a regularization

term (Kullback-Leibler divergence) to ensure latent space follows a prior distribution (usually Gaussian)

2 3D Data Representation

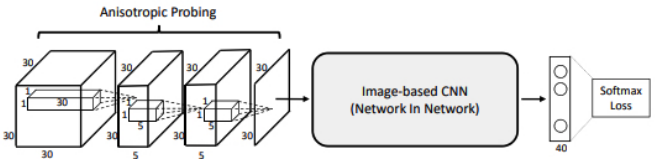
- Explicit Representations: Voxel grids, Point Clouds, Polygonal Meshes
- Implicit Representations: Signed Distance Functions (SDFs), Occupancy Networks

2.1 Volumetric Grids

Volumetric Data Structures

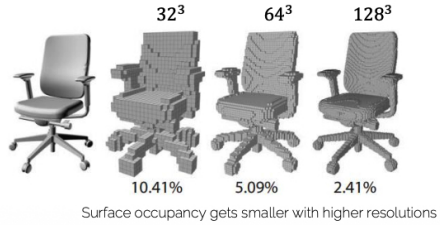
- Occupancy grids
- Ternary grids (2-D triangular coordinate system, e.g., Barycentric coordinate)
- (Signed) Distance fields

Extension of AlexNet/2D CNNs to 3D CNNs by using 3D convolutional kernels: Object Classification with 3DCNNs, 2016



Summary:

+ Simple data structure
+ Encode free space and distance fields
+ Naturally extend 2D CNNs and other concepts
+ Faster training due to an additional dimension per sample (but often lack 3D data)
– High memory consumption (cubic growth), a lot is used for empty space the higher the resolution
– Require a lot of processing time (use sliding window or fully-convolutions)



Surface occupancy gets smaller with higher resolutions

2.2 Volumetric Hierarchies

Octrees

- Hierarchical data structure that recursively subdivides 3D space into eight octants (like a quadtree in 2D).
- 3D partitioning: higher resolution at surfaces, lower resolution in empty space for applying 3DCNN kernels
- Can be used for discriminative tasks (like object classification, segmentation) or generative tasks (like 3D reconstruction).

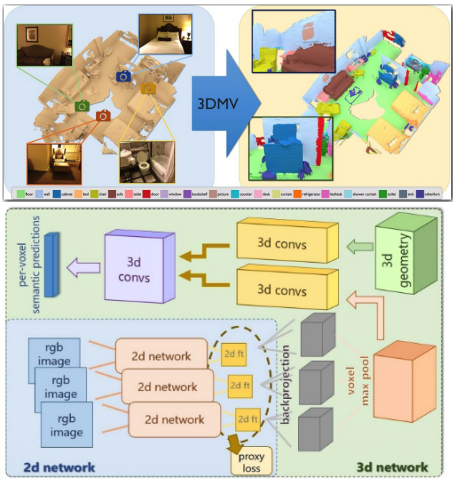
Summary:

+ Great for reducing memory and runtime
+ Increases strongly performance of 3D CNNs
– Easier for discriminative tasks when structure is known, more complex for generative tasks (split voxel that are partially occupied)

2.3 Hybrid: Volumetric + Multi-View

- Combine volumetric representation with multi-view images (colors) to improve segmentation
- Separate 2D (image) and 3D (voxel) feature extraction + back-projection of 2D features into 3D space

3DMV: 3D Voxel + Multi View Semantic Segmentation, 2016



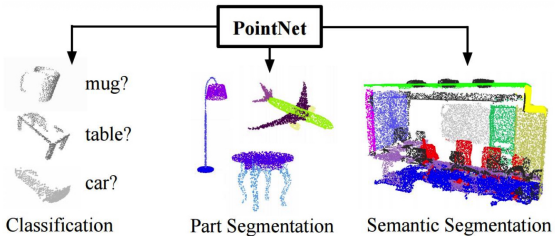
Summary:

+ Nice way to combine color and geometry
+ Good performance
– End-to-End training helps less than expected
– Could be faster

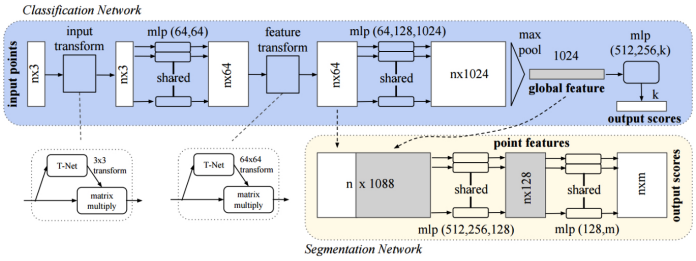
2.4 Point Clouds

- Unordered set of 3D points (often from LiDAR or depth sensors)
- Each point can have additional features (e.g., color, intensity, normals)
- Direct processing of point clouds using specialized neural networks (e.g., PointNet, PointNet++)

PointNet: 3D Classification and Segmentation for Point Clouds, 2017



- take n points as input
- apply feature transformations and aggregate features by max pooling
- classification net: output scores for k classes
- segmentation net: concatenate global and local features for per-point classification



PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, 2017

- learn hierarchical representation of point clouds
- apply multiple PointNets at different locations and scales
- multiscale grouping (MSG) and multi-resolution grouping (MRG) for local features

Point Convolutions

- Transform points to continuous representation using radial basis functions (RBFs) or kernel density estimation (KDE)
- Apply convolutional operations on the continuous representation

Point Transformer

- Use self-attention mechanisms to capture relationships between points
- Learn point features by attending to neighboring points

2.5 Sparse Convolutional Networks

- Efficiently process sparse 3D data (e.g., point clouds, voxels with many empty spaces)
- Use sparse convolutional operations that only compute on non-empty voxels
- To improve efficiency, use sparse hashes (e.g. QSParseConv, MinkowskiEngine) instead of masking conv output
- Latency on GPU for hash lookup is hidden by parallelism

Summary:

+ Efficient representation, only store surface points
+ Fast training and testing
+ Cover large space in one shot
– Can not represent free space
– Perform worse than volumetric methods in some tasks (a lot ongoing research)

2.6 Polygonal Meshes

- Represent 3D surfaces using vertices, edges, and faces (typically triangles or quadrilaterals)
- Widely used in computer graphics and 3D modeling
- Process via graph neural networks
  - Message Passing
  - Graph Convolutions
  - Transformers
- Process via specialized mesh convolutional networks
  - MeshCNN
  - Geodesic CNNs
  - Spectral CNNs

2.7 Signed Distance Functions (SDFs)

2.8 Occupancy Networks

3 3D Datasets

3.1 3D Shapes

ShapeNet

- Main Dataset, synthetic 3D models
- 51.3k shapes, 55 classes
- mostly chairs, mediocre textures

Objaverse-XL

- 10mio 3D shapes
- heterogeneous distributed

3.2 3D Scenes

3D-FRONT

- furnished rooms with layouts and semantics
- 18k rooms, 7k furniture objects

ScanNet

- Kinect-style reconstructions of indoor scenes
- 1.5k scenes, 2.5 million views
- Semantic and instance annotations
- from TUM, with standardized evaluation benchmark

4 Neural Fields

- Field: quantity defined for all spatial or temporal coordinates
- Neural Field: represent a field represented by a neural network

4.1 MLP as Data Structure

- MLP can represent complex signals (images, 3D shapes, etc)
- Sparse encoding of signal, shifts capacity where it needs it (based on optimization)
- Can be used for compression, denoising, super-resolution, etc.
- Often also referred to as Implicit Neural Representation (INR) or Coordinate-based MLP

4.2 Implicit vs Explicit

Implicit Neural Representation: Misleading since only the data structure is implicit not the function itself.

$$\text{Explicit function:} \quad f(x) = y \tag{1}$$

$$\text{Implicit function:} \quad x^2 + y^2 - 1 = 0 \tag{2}$$

Signed Distance Field (SDF) A signed distance field assigns to each point $x \in \mathbb{R}^n$ a real value that represents the distance to the closest surface, with the sign indicating whether the point lies inside or outside the surface:

$$\mathrm{SDF}(x) = \begin{cases} < 0, & \text{inside surface,} \\ = 0, & \text{on surface,} \\ > 0, & \text{outside surface.} \end{cases}$$

Occupancy Field An occupancy field assigns to each point $x \in \mathbb{R}^n$ a binary indication of whether the point lies inside the sur-

face:

$$\mathrm{Occ}(x) = \begin{cases} 1, & \text{if } \mathrm{SDF}(x) \leq 0, \\ 0, & \text{if } \mathrm{SDF}(x) > 0. \end{cases}$$

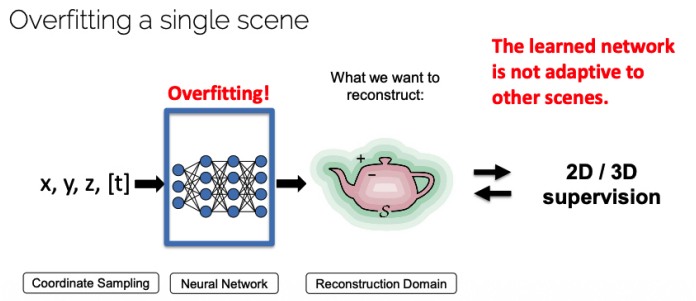
Key Difference

- **SDF:** continuous-valued; encodes signed distance from the surface.
- **Occupancy:** binary-valued; encodes only inside/outside.

4.3 Overfitting vs. Generalization

- Overfitting as an artifact of neural fields: MLPs can memorize training data without generalizing.
- Overfitting as a debugging tool: helps to verify that the network architecture and training procedure are functioning correctly.
- Overfitting as a goal: to represent a specific signal or shape accurately.

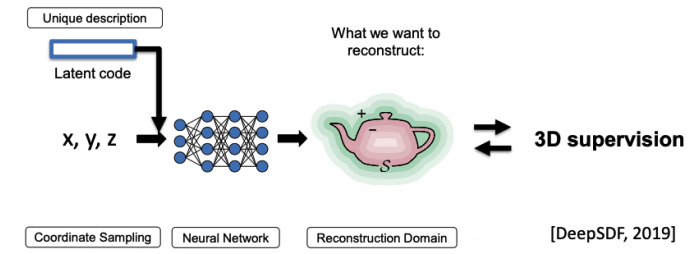
Overfitting Single Scene



- Overfitting a single scene from multiple views
- Input set of images and their camera parameters

Overfitting Multiple Scenes

- Each scene should have a unique description
- To prevent latent code bias in activations, either a positional encoding is required or a learned latent code (optimized during training)
- learned latent code: has semantic meaning: like parametrization of Radial Basis Functions (RBFs)
- Changing latent code allows interpolation between data



4.4 Inductive Bias

Shift invariance is very important to account for bias in activation based on position/coordinate values

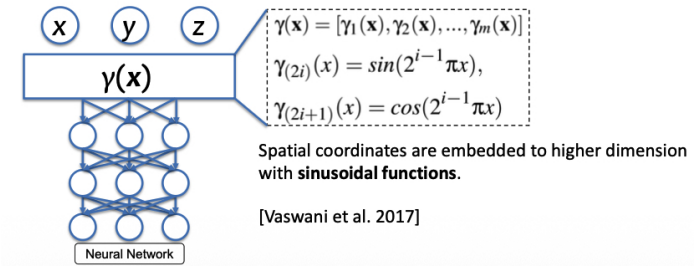
- Positional encoding (Sinusoidal high-dimensional mapping of coordinates)
- SIREN Activations (sinusoidal activations to represent high-frequency functions)

On the Spectral Bias of Neural Networks, 2018

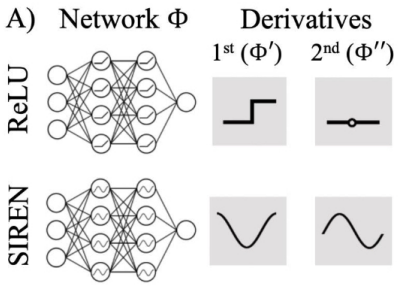
Positional Encoding Sinusoidal / learned positional encodings allow:

- smooth distance representation,
- easy extrapolation,
- relative comparison (difference of phases encodes distance).

- This provides a useful inductive bias: tokens that are close in sequence have similar positional embeddings.
- Raw integer positions do *not* provide this similarity.

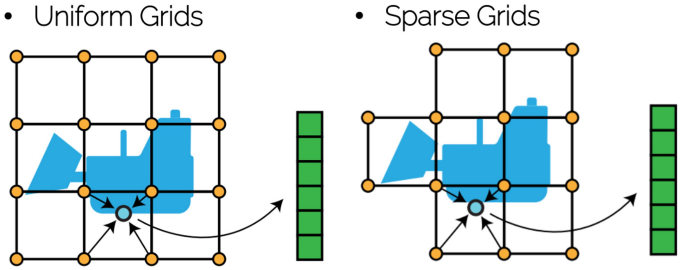


Activation Functions SIREN uses sinusoidal activations to fit to high-frequency functions.



4.5 Hybrid Representations

- **Uniform grid**
  - Dense, regularly–sampled grid over the whole domain
  - Easy indexing and interpolation
  - Memory and computation scale poorly with resolution
  - Wastes resources in empty or simple regions
- **Sparse grid**
  - Only stores cells where signal / geometry is complex
  - Greatly reduces memory and compute, O(log(n)) access time
  - GPU compatible data structure
  - Established operations like sparse 3D convs
  - Requires hierarchical / hashed data structures
  - More complex to implement and query



4.6 Summary

1. Overfit an MLP
2. Positional Encoding: prevent inductive bias from coordinates
3. Activation Functions: non-linearities to represent high-frequency functions

- MLP: fully connected layers, connect everything with everything
- CNN: local connections, weight sharing, translation equivariance
- Transformer: learn which connections are important

5 Neural Radiance Fields (NeRF)

- Neural Radiance Field (NeRF) is a neural network-based representation for 3D scenes that encodes both geometry and appearance.
- Enables photo-realistic novel view sybnthesis from a sparse set of input images.
- Now-standard approach to neural volumetric rendering.

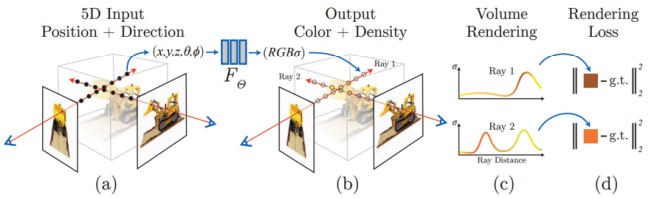
Neural Radiance Fields (NeRF), Mildenhall et al., ECCV 2020

5.1 Core Idea

- NeRF represents a scene as a continuous 5D function:

$$F_\theta : (x, y, z, \theta, \phi) \rightarrow (\sigma, \mathbf{c})$$

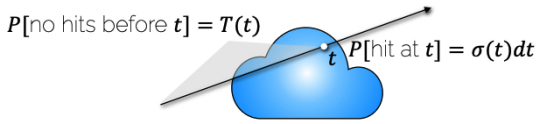
- 5D Input:
  - (x, y, z): 3D position
  - (θ, ϕ): viewing direction
- Outputs (MLP):
  - $\mathbf{c}$: emitted color at sample point perceived from direction
  - σ: volume density (chance of absorption at specific location)
- Volumetric rendering allows to reconstruct the final image using sampled points along camera rays.
- All steps are end-to-end differentiable, allowing training via gradient descent.



- Neural: Use a Neural Network as scene representation instead of explicit data structures.
- Volumetric: continuous, differentiable rendering model without concrete surface representation (similar to a cloud of particles).
- Rendering: compute color along rays through 3D space.

5.2 Probabilistic Interpretation

- Ray traveling through scene hits a particle at distance t, where color $c(t)$ returned.
- Probability of hitting a particle at small interval dt at distance t: $\sigma(t)dt$
- Probability of not hitting any particle before distance t (Transmittance): $T(t)$



The product of these probabilities tells us how much you see the particles at t:
P[first hit at t] = P[*no hit before* t] × P[hit at t]
$= T(t)\sigma(t)dt$

- P[no hit before t + dt] = P[no hit before t] * P[no hit at t]
- $T(t + dt) = T(t) * (1 - \sigma(t)dt)$

$$T(t + dt) = T(t)\big(1 - \sigma(t)\, dt\big)$$

Solve via Taylor Expansion:

$$T(t + dt) \approx T(t) + T'(t)\, dt = T(t) - T(t)\sigma(t)\, dt$$

$$\frac{T'(t)}{T(t)}\, dt = -\sigma(t)\, dt$$

$$\int_{t_0}^{t} \frac{T'(s)}{T(s)}\, ds = -\int_{t_0}^{t} \sigma(s)\, ds$$

$$\log T(t) = -\int_{t_0}^{t} \sigma(s)\, ds$$

$$T(t) = \exp\left(-\int_{t_0}^{t} \sigma(s)\, ds\right)$$

This gives P[first hit at t]:

$$p(t) = T(t)\ * \sigma(t)dt$$

To get the expected color returned by the ray, we need to integrate over all sampling points t_i along the ray:

$$C = \int_{t_0}^{t_n} T(t)\, \sigma(t)\, c(t)\, dt$$

Note the nested integral, since $T(t)$ itself is an integral over density $\sigma(s)$.

5.3 Training

- NeRF is trained via supervised inverse rendering using ground-truth images.
- Loss:

$$\mathcal{L} = \sum_{\text{rays}} \|C_{\text{pred}} - C_{\text{gt}}\|_2^2$$

- Requires known camera poses (typically estimated with COLMAP).

5.4 Advantages

- Continuous, high-resolution scene representation.
- High-quality novel-view synthesis with realistic lighting and specularities.
- Compact representation: scene stored in MLP weights.

5.5 Limitations (Original NeRF)

- Slow training and rendering due to MLP evaluations on every ray sample.
- Static scenes only.
- Requires accurate camera pose estimation.