0.1 Glossary

- Image classification: assign a class label to an image
- Object localization: draw a bounding box around one or more objects in an image
- Object detection: draw bounding box and assign a label
- Object segmentation
- Object recognition

0.2 Datasets

- CalTech-101
- VOC-2012
- ILSVRC-2013
- ImageNet

0.3 Convolutional Networks

- Convolution layers
- Pooling layers
- Fully-connected layers
- Filters
- Stride
- BatchNorm
- Activation functions

0.4 CNN Architectures

- ImageNet Classification Challenge
- AlexNet
  - 227 x 227 inputs
  - 5 convolutional layers
  - Max pooling
  - 3 fully-connected layers
  - ReLU nonlinearities
  - 61 million parameters total
  - Most memory usage is in the early convolution layers

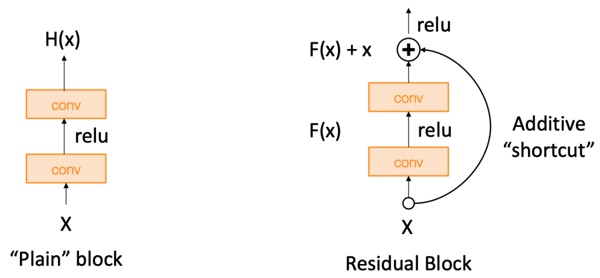
- Nearly all the parameters are from the fully-connected layers
- Most floating-point ops occur in convolution layer

- VGG
  - Enabled training deeper networks
  - Established standard network design patterns
  - Design rules:
    * All conv are 3x3 stride 1 pad 1
    * All max pool are 2x2 stride 2
    * After pool, double the # of channels
  - Notes: 2 3x3 conv has the same receptive field as a single 5x5 conv, but fewer parameters and less computation!
  - VGG-16 total: 138 million parameters

- GoogleLeNet
  - Many innovations for efficiency: reduce parameters, memory usage, and computation
  - Stem network: aggressively downsample input from the start
  - Inception module: local unit with parallel branches, use 1x1 bottleneck to reduce channel before conv
  - No FC layers at the end
  - Uses global average pooling to collapse spatial dimensions and one linear layer
  - Hack to overcome deepness of network, use "auxillary classifier" at intermediate points in network
  - This work was before BatchNorm!

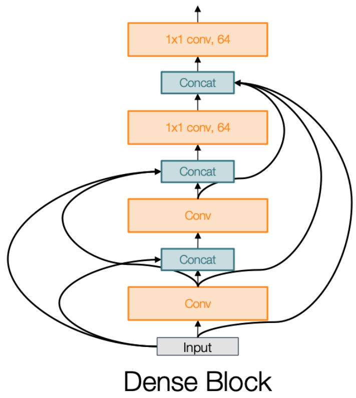
- ResNet
  - 152 layers!
  - Change network so learning identity function with extra layers is easy
  - Able to train very deep networks

Figure 1: Residual Block



- Densely Connected Neural Networks
  - Each layer is connected to every other layer in feedforward

Figure 2: Dense Block



  -

- Tiny networks
  - MobileNet
  - ShuffleNet
- Neural Architecture Search
  - Automate searching for efficient network architectures
  - A network outputs network architectures
  - The controller generates child networks and trains them
  - After training a batch of child networks, make gradient step on controller network
  - VERY EXPENSIVE! Trained 800 GPUs for 28 days

0.5 Recurrent Networks

- Recurrent Neural Networks
  - useful for processing sequences with inputs
  - one-to-one, one-to-many (image captioning), many-to-one (video classification), many-to-many (machine translation)
  - Key idea is that RNN have an "internal state" that is updated as a sequence is processed

$$h_t = f_W(h_{t-1}, x_t)$$

$$\begin{aligned} x_t &= \text{input vector} \\ h_t &= \text{new state} \\ f_W &= \text{function with parameter W} \end{aligned}$$

- Vanilla RNN

$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\
y_t &= W_{hy}h_t + b_y
\end{aligned}
$$

- Reuse weight matrix at every timestep
- Backpropagation through time
- RNN forwards the entire sequence to compute loss and then backward through the entire sequence to compute gradients
- Language modeling - generating Shakespeare poems, generating code, etc
- Image captioning - first run CNN on image and feed output from FC-4096 into the RNN along with captions
- Backprop results in exploding / vanishing gradients
- For exploding gradients, we can use gradient clipping to scale the gradient if norm is too big
- For vanishing gradients, we need to change the RNN architecture

- Long-short Term Memory

$$
\begin{aligned}
i_t &= \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\
f_t &= \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\
o_t &= \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\
g_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

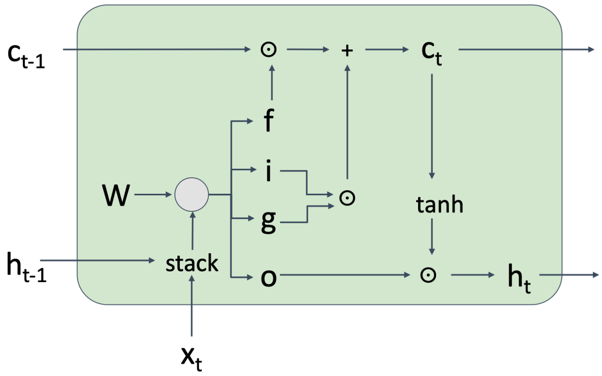
$$
\begin{aligned}
i &= \text{input gate: whether to write to cell} \\
f &= \text{forget gate: whether to erase cell} \\
o &= \text{output gate: how much to reveal cell} \\
g &= \text{gate gate (?): how much to write in a cell}
\end{aligned}
$$

- Backpropagation from c_t to c_{t-1} is only elementwise multiplication by f, does not depend on matrix multiply by W
- Gated Recurrent Units
  -

0.6 RCNN and variants

- **[rcnn]**
- Region Proposal

Figure 3: LSTM Cell



  - Generate and extract category independent region proposals, candidate bounding boxes
  - Use CV technique called "selective search"
  - AlexNet

- Feature Extractor
  - Extract feature from each candidate region
- Classifier
  - Classify feature as one of the know classes

0.6.1 Other popular architectures

0.7 Detection

0.7.1 Single-stage detectors

0.7.2 Two-stage detectors

0.8 Segmentation

0.8.1 Semantic segmentation

0.8.2 Instance segmentation

0.8.3 Keypoint estimation

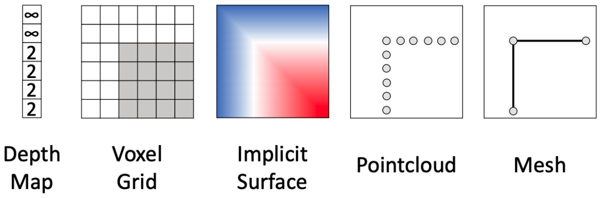
0.9 Domain Adaptation

0.10 3D

0.10.1 Representations

- Depth Map

Figure 4: 3D shape representations



  - Distance from camera to object into the world at each pixel
  - Often combined with RGB image = RGB-D image (2.5D)
  - Can't capture occuluded areas
  - e.g. Microsoft Kinect and new IPhone-12 sensors
  - Task: predicting depth map from RGB image
  - Scale / depth ambiguity
- Surface Normals
  - Normal vector to object in world at pixel
  - Useful for robotics applications such as grasping objects
  - Can get ground-truth surface normal using synthetic data or multiview 3D reconstruction
  - Predict surface normal representation, loss: penalizes the angle between vectors
- Voxel Grid
  - A shape is a V x V x V grid of binary occupancies, any type of data can be stored in a cell of voxel representation
  - Like segmentation mask in 3D rather than 2D
  - Need high spatial resolution to capture fine-grain objects
  - Scaling to high resolutions is nontrivial, computationally expensive
  - Process these with 3D convolution
  - Generating voxel shapes using 3D convolution, flatten 2D features and convert into 3D feature
  - Predict per voxel and take loss per voxel
  - Memory usage scales cubically, not tractable
  - Oct-Trees for scaling voxels with heterogenous resolutions
- Implicit Surfaces
  - Learn function to classify arbitrary 3D points as inside / outside the shape or predict some sort of data
  - The surface of the 3D object is the level set

    - $o : \mathbb{R}^3 \to \{0, 1\}$
- Pointclouds
    - Represent shape as set of points in 3D space
    - Can represent fine structures without huge # of points
    - Requires new architectures / losses
    - Don't explicitly represent the surface! Need some postprocessing to get a mesh representation
    - PointNet: run MLP on each point in the cloud to generate per-point feature vector → max pooling then a fully-connected vector
- Triangle Mesh
    - Commonly used in computer graphics
    - Represent 3D shape as a set of triangles
    - Adaptive: can reprsent flat surfaces very efficiently, allocate more faces with fine detail
    - Can interpolate data on vertices over whole surfaces like RGB color, normal vector
    - Pixel2Mesh: single RGB image → triangle mesh

0.10.2 Depth estimation

0.10.3 3D shape prediction

0.10.4 Voxels and pointclouds

0.10.5 Structure from Motion

0.10.6 View Synthesis

0.10.7 Differentiable Graphics

0.11 Dataset biases

0.12 Vision and language

0.13 Vision and sound

0.14 Attention for vision