1 NN, CNN

1.1 Perceptron Learning Algorithm (PLA)

1. Decision rule: $\text{sign}(w^T x) \neq y$.
2. Update rule:

$$w_{t+1} = w_t + y_i x_i$$

3. PLA works when the data is linearly separable.

1.2 CNN

$$L_{\text{out}} = \frac{L_{\text{in}} + 2 \cdot \text{P} - \text{dilation} \cdot (K-1) - 1}{\text{S}} + 1$$

1.3 Neural Network Optimization

1.3.1 Loss Functions

Multi-class SVM loss:

$$L = \lambda \|w\|^2 + \sum_i \max(0, 1 - w^T x_i y_i)$$

Probability distribution (softmax):

$$L = \frac{e^{w^T x_j}}{\sum_k e^{w^T x_k}}$$

Negative log-likelihood:

$$L = \lambda \|w\|^2 + \sum_k -\log \frac{e^{w^T x_k}}{\sum_k e^{w^T x_k}}$$

1.3.2 Optimization Techniques

1. **Grid Search:** bad for high dim $O(\text{samples per dimension}^{dim})$.
2. **Random Search:** Simple and effective in high-dimensional spaces but does not guarantee finding the global optimum.
3. **Gradient-Based Methods:**

$$w = w - \eta \nabla_w L(w)$$

1) Follow the opposite direction of the gradient.
2) Subgradient: can be applied to convex, non-differentiable functions. For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, a vector $g \in \mathbb{R}^n$ is called a **subgradient** of f at point x if the following inequality holds for all $y \in \mathbb{R}^n$:

$$f(y) \geq f(x) + g^\top (y - x)$$

1.4 Gradient Descent

1. **Vanilla/Batech GD:** all data
2. **Stochastic:** single data pt per update.
3. **Mini-batch:** random subsets of data.

1.5 Underfitting and Overfitting

1. **Underfitting:** Poor on train/val due to high bias. 2. **Overfitting:** Good on train, poor on val due to high variance. 3. **Solutions:** a. Remove redundant data. b. Avoid large changes to the model.

2 Object Detection (OD)

1. Predict bbox labels and confidence.
Localization: Sliding win.
Classification: Often SVM.

2.1 Eval Metrics

$$\text{IoU} = \frac{\text{intersection}}{\text{union}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

4. **mAP:** Mean AP over all classes.

2.2 Sliding Window-Based OD

1. Sliding windows over multi scales/sizes.
2. **Challenges:** Inefficient for large-scale datasets.
3. **Human Detection w/ HOG:** a. Train pedestrian template w/ SVM.
b. Apply template across img.
c. Use local maxima of response for detection.
d. Multi-scale detection w/ HOG pyramid.

2.3 Discriminative Part-Based Models (DPM)

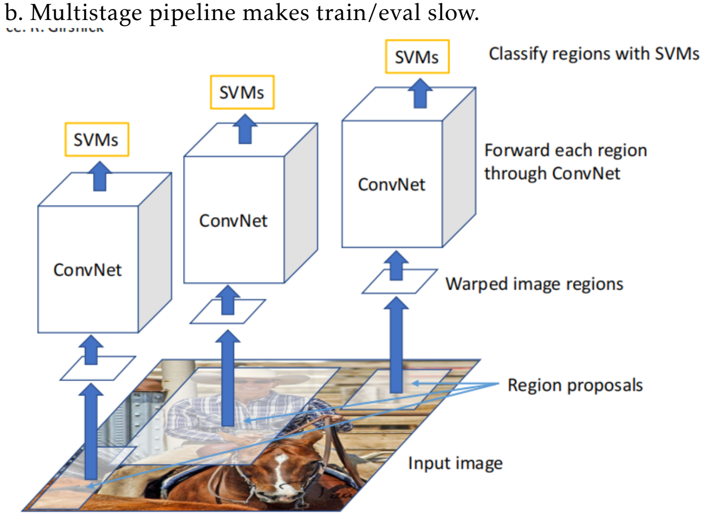
1. Objects are decomposed into parts.
2. Example: Deformable Part-based Models (DPM).
a. Detect parts and combine using an ensemble.
b. 'Springs': spatial connections between parts.

2.4 Proposal-Based OD

1. **Selective Search:** Uses diverse cues to merge superpixels into regions for better bbox proposals.
2. **EdgeBoxes:** a. Exploits edge info w/ a trained edge detector. b. Forms object-like groupings (objects tend to be enclosed by edges).

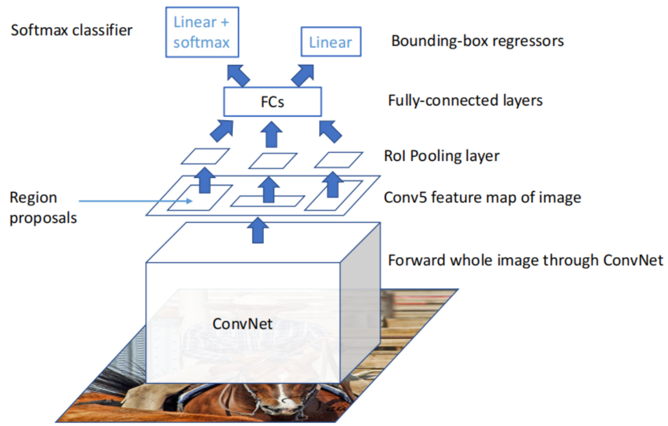
2.4.1 R-CNN (Region-Based CNN)

1. Region proposals + CNN features. 2. **Pipeline:** a. Regions generated using Selective Search.
b. Features extracted w/ AlexNet.
c. Final detector: Non-max suppression (NMS) + linear SVM.
d. Bbox regression refines proposals.
3. **Pros:** High detection accuracy.
4. **Cons:** a. Not end-to-end trainable.
b. Multistage pipeline makes train/eval slow.



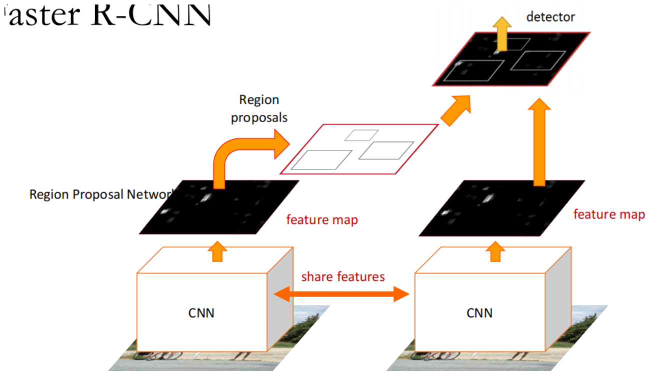
2.4.2 Fast R-CNN

1. Uses shared feature map from CNN.
a. In R-CNN, a separate CNN is trained for each proposal.
b. Fast R-CNN fixes this by using a single shared feature map.
2. **ROI Pooling:** Region proposals are mapped to the shared feature map. Regions are resized to the same size through ROI pooling.
3. Two losses:
a. Softmax loss for classification.
b. Bbox regression loss.



2.4.3 Faster R-CNN

1. Adds a Region Proposal Network (RPN) to the pipeline.
2. **Key Components:**
a. Fully conv net:
i. Input: Shared feature map.
ii. Output: Region proposals.
b. Fixed-size window over conv layers.
i. Predict obj / no obj for proposals.
ii. Regress bbox coords w.r.t. anchors.
3. 2 classification loss and 2 bbox regression loss.



2.4.4 YOLO (You Only Look Once)

1. Combines classification and regression.
2. **Pipeline:**
a. Image divided into $S \times S$ grids.
b. For each grid, predict:
i. Bboxes.

ii. Confidence scores.
iii. Class probs.
c. Box filtering w/ NMS.
3. **Pros:**
a. Real-time detection.
b. Captures global context.
c. Simple design.
4. 3 Losses: Localization loss, confidence loss, classification loss.

3 Image Segmentation

3.1 Datasets and Metrics

1. **Datasets:** PASCAL VOC, COCO, Cityscapes, etc.
a. Positive: Foreground.
b. Negative: Background.
2. **Metrics:**
a.

$$\text{Pixel Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

b.

$$\text{IoU(Jaccard Index)} = \frac{\text{TP}}{\text{TP+FP+FN}}$$

c. mIoU (Mean IoU for Multi-Class)
d. F1 Score (DICE):

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \text{DICE} = \frac{2 \cdot \text{Prediction} \cap \text{GT}}{\text{Prediction} + \text{GT}}$$

$$DICE = F1 = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

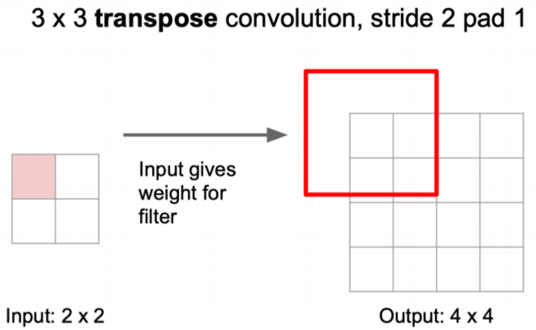
Note: DICE same as F1 score in pixel seg.
F score: Closer to avg performance. IoU: Worst-case performance.

3.2 Approaches to Segmentation

1. **Traditional:** Cluster based on color/position.
2. **Sliding Windows:** Inefficient.
3. **End-to-End Fully Conv Network (FCN):**
a. Downsample
b. Upsample:
i. Nearest Neighbors
ii. "Bed of nails"
iii. Max unpooling: uses max pos from pooling layers.
iv. Transpose Conv: Learnable.

$$L_{\text{out}} = (\text{L}_{\text{in}} - 1) \cdot S + K - 2P$$

If pad = 1, the 1st col/row will be removed in the output.



4. **U-Net:** i. Skip connections for low (high res) and high (low res) levels.
ii. Fuse using concatenation.

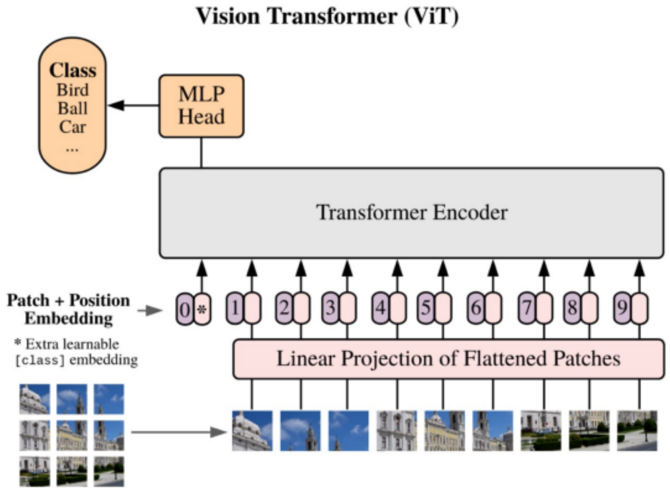
4 Transformers

1. Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

2. Positional Embedding: High-freq sin / cos waves.

3. Vision Transformer (ViT):
a. Patchify & embedding: Images are divided into 16×16 patches.
b. Learnable token [CLS] is prepended to every seq of patches.
c. Classifier token is in the first $1 \times D$ row.
d. Positional embeddings (PE) are learnable.



4. Compact Transformers:
a. **Sequence pooling:**
1) Instead of adding [CLS] to seq, add it as an extra $D \times 1$ layer after encoder.
2) Get Nx1 vec from NxD vec.
3) Apply softmax to vec and mul w NxD vec → 1xD vec → classifier.
b. Patchify + embedding ≈ conv with overlapping convolutions.

5. Examples:
a. ViT-lite: Fewer layers/heads, smaller imgs.
b. Compact Vision Transformer (CVT): Seq pool + conv layers.
c. Convolutional Transformer (CCT): Less sensitive to PE due to convs.

6. Add Attention to Vision:
a. Feed imgs to transformers.
b. CNN + attention/self-attention layers.
c. Replace CNN w/ attention or transformers (e.g., Swin Transformer).

5 Diffusion Model

1. **Forward Process (+ Noise):**

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot I)$$

Adding noise at each time step.

2. **Backward Process (Denoising):**

$$p(x_{t-1}|x_t) \propto p_0(x_t|x_{t+1}) \prod_{t=1}^{T} q(x_t|x_{t-1})$$

$$p_0(x_{t+1}, x_t) = \mathcal{N}(x_t; \mu(x_t, t), \sigma(x_t, t))$$

Iteratively denoising until the original data is reconstructed.
3. **Generative Models:**
a. **VAE:** Maximize the variational lower bound.
b. **GAN:** Adversarial training.
c. **Diffusion Model:** Add noise and then remove it.

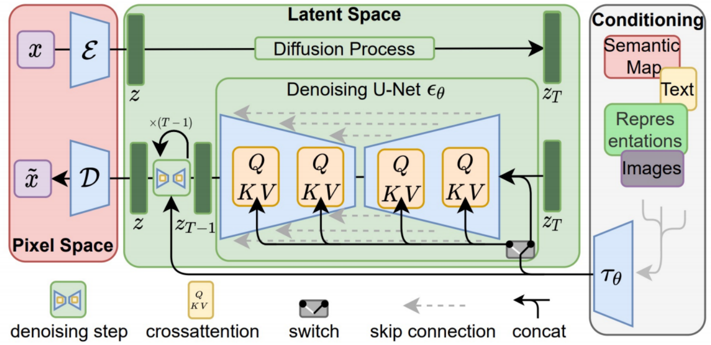
4. **Time Embedding for DDPM-UNet:** fed to each layer.

5. **Score-Based SDE Perspective:**
a. Forward process as a stochastic differential equation:

$$x_t = x_{t-1} - \frac{\beta(t)}{2} \cdot \Delta t + \sqrt{\beta(t)} \cdot \mathcal{N}(0, I)$$

$$dx = -\frac{\partial \beta(t)}{\partial x} \cdot \Delta t + \sqrt{\beta(t)} \cdot dw$$

6. **Latent Diffusion Model (Stable Diffusion):**
a. LDM: Autoencoder + latent diffusion model (U-Nets) + conditional encoders.
b. **Applications:**
Image generation, Text-to-image generation, Semantic synthesis, Super-resolution, Inpainting.



7. **Video Diffusion Model:**
a. Direct training on video (e.g., Video LDM).
b. Pretrained text-to-image model (e.g., Text2Video-Zero).
c. Pretrained model + training on video motion (e.g., Animate-Diff).

6 Large Multimodal Models (MLLM)

1. **Tokenizer:** Converts input sequences into discrete tokens.
2. **Next Token Prediction:**
a. Causal mask applied to self-attention.
b. Classifier predicts the next token from hidden states of previous tokens.
3. **Model Training:**
a. Pretraining.
b. Instruction tuning.
c. Preference alignment.