**DINO:**

* Teacher student models
* Both gets separate pov’s (cropped version) of same input image
* Student tries to match teacher’s output and has gradient learning and ground truth is teacher’s output.
  + Teacher’s version is global (has more than 50% image)
  + Student’s version is local (has less than 50% image)
  + Therefore, teacher should have better understanding thus better output
* Teacher learns by ema (exp moving average) of student... it learns very slowly
* **Loopholes covered**
  + Model can try to predict uniform weitage across classes regardless of input
    - Sharpening: To avoid this, temp Ts, Tt is applied before softmax to sharpen the logits.
  + Both can agree on same label and keep predicting (student will learn nothing as teacher is also wrong)
    - Centred output: subtract the mean from teacher’s logits.

**SAM:**

* Three components
  + an image encoder
    - run once and used with different prompts on same image
  + a flexible prompt encoder
    - can be point, box, text
  + a fast mask decoder
    - combines prompt and image embeddings to predict masks

**R-CNN**

* Selective Search (select 2000 proposed regions, where objects can be...)
  + **superpixel segmentation algorithm:** Superpixels are groups of neighbouring pixels with similar colour or texture.
  + For each segmented region, extract the following features: ● Color histogram (RGB or HSV) ● Texture histogram (gradient-based) ● Size of the region ● Shape / bounding box
  + Start merging the most similar neighboring regions based on a similarity function. Continue merging until the whole image becomes one region.
* Resize each region to all regions of same size (224\*224) as CNN can process fixed size regions
* Extract features via CNN
* Predict labels of regions via SVM
* trains a regressor (a small linear model) to fine-tune the position and size of each bounding box to better fit the object.

**FAST R-CNN**

* Instead of feeding 2000 cropped regions through the CNN one at a time, run the CNN once on the whole image, *then extract features for all regions from a shared feature map (RoI)*.
* It solves the speed and inefficiency problems of R-CNN by sharing computation and combining classification and bounding box regression into a single deep network.
  + Two output branches
    - Classification (cross entropy loss)
    - Bounding box regression (smooth L1 loss)

**YOLO**

* So instead of: ● “Is there a dog in this box?” → classification ● “Where is the box?” → regression YOLO says: ● “Given this grid cell, regress the entire set of outputs (coordinates + class probs) from the image directly.” → unified regression
* **Confidence (objectness)** = P(object) × IOU(predicted\_box, ground\_truth\_box)
  + P(object) is the probability that there’s any object in the box.
  + IOU is how well the predicted box overlaps the actual object.
  + So even if there's an object, if the box is off, the confidence will be low.
* **Class Probabilities:** These are the conditional probabilities of each class given that there is an object:
* Bounding box dimensions (x, y, height, width)
* Confidence score (weather there is an object and the bounding box is how much correct, if the object is there but the box is not correct, confidence score will still be low)
* Class probabilities, prob of each class if there is a object in bounding box

**YOLOv2**

* Anchor boxes: Run k-means clustering on the training set bounding boxes to find k anchor boxes.
  + Each anchor box has a fixed width and height (w\_anchor, h\_anchor).
  + **shape**
    - Grid size: 13x13, Anchor boxes: 5,
    - Output tensor shape: 13 x 13 x (5 x (5 + num\_classes)),
    - Each of the 5 boxes predicts 5 values + class probs: ■ (tx, ty, tw, th, objectness) + class\_probs
  + Ap (average precision)
  + Iou (intersection/union)

**Diffusion**

* Training data is added noise in T timestamps
* It is reversed (by learned nn) to detect the added noise from timestamp t-1. And learn accordingly
  + Markov chain: image at timestamp t depends only on timestamp t-1

**Stable Diffusion**

* Image is first compressed into latent
* Then latent is denoised
* Then latent is converted again into image

**CLIP**

* Image generation through text prompt via diffusion models
* Denoising is conditioned via text prompt embeddings (cross attention) in UNET
* Q: image feature-map, K,V: from text embeddings
* image features "query" the text embeddings to decide what to pay attention to!
* Noisy Feature Maps (Query) ---> Cross Attention ---> Updated Feature Maps
* Text Embedding (Key & Value)