Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Maximilian Bernhard, Niklas Strauß

Munich, 02.12.2024

Deep Learning and Artificial Intelligence

WS 2024/25

Exercise 7: Computer Vision Architectures

Exercise 7-1 Backbone Architectures

(a) On a high level, the residual blocks of a Residual Network (ResNet) can be described as

$$x_{l+1} = f_l(x_l) + x_l,$$

where l is the layer index, $f_l(\cdot)$ is the l-th layer and x_l is the feature map at the l-th layer. Express the final feature map x_L as a function depending on x_l for any $l = 1, ..., L-1$.

Note: You do not have to write the term in a way that it solely depends on x_l, but it may still also depend on the intermediate terms $x_i, i = l, ..., L-1$.

Detailed Solution

$$\begin{aligned} x_L &= f_{L-1}(x_{L-1}) + x_{L-1} \\ &= f_{L-1}(x_{L-1}) + f_{L-2}(x_{L-2}) + x_{L-2} \\ &= \cdots \\ &= x_l + \sum_{i=l}^{L-1} f_i(x_i) \end{aligned}$$

(b) Based on the previous result, derive an expression for the gradient from the loss $\mathcal{L}$ w.r.t. the feature x_l in a residual network, i.e., $\frac{\partial \mathcal{L}}{\partial x_l}$. Interpret the result.

Detailed Solution

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_l} &= \frac{\partial \mathcal{L}}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} \\ &= \frac{\partial \mathcal{L}}{\partial x_L} \cdot \frac{\partial \left(x_l + \sum_{i=l}^{L-1} f_i(x_i) \right)}{\partial x_l} \\ &= \frac{\partial \mathcal{L}}{\partial x_L} \cdot \left(1 + \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} f_i(x_i) \right) \\ &= \frac{\partial \mathcal{L}}{\partial x_L} + \frac{\partial \mathcal{L}}{\partial x_L} \frac{\partial}{\partial x_l} \sum_{i=l}^{L-1} f_i(x_i) \end{aligned}$$

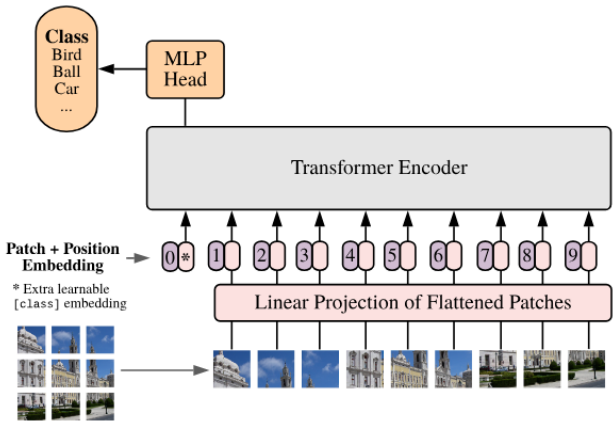

Figure 1: ViT architecture. Source: `https://arxiv.org/pdf/2010.11929`.

This means that the gradient that comes directly from the loss $\frac{\partial \mathcal{L}}{\partial x_L}$ is propagated to x_l without modification. This is desirable as it ensures that we have useful gradients for all x_l, while gradients propagated through many layers may suffer from vanishing or exploding.

(c) In image classification, we need to reduce a spatial feature map to a fixed-size vector at some point in order to eventually arrive at logit scores representing the whole image. Compare how this is achieved in ResNet and Vision Transformer (ViT).

Detailed Solution

In ResNets, this is done by global average pooling, i.e., given the feature map x, we compute the fixed-size vector x_{cls} for classification as

$$x_{cls} = \frac{\sum_{(i,j)\in[H]\times[W]} x_{ij}}{H \cdot W}.$$

In contrast, ViT uses an attention-based approach where a dedicated classification token is fed into the model next to all patch tokens, so that it can aggregate information from all other tokens via self-attention. The initial values of the classification token/embedding are learned just like any other network parameter.

After these aggregation techniques, the obtained vectors representing the whole image are fed to a classification head (MLP) in both cases.

Note that there are also implementations of ViT that use an average pooled feature vector instead of a dedicated token for classification.

(d) Compare ResNet, ViT, and Swin regarding their inductive bias for image data.

Detailed Solution

ResNet as a CNN exhibits the most amount of inductive bias as their convolutional design explicitly emphasizes the importance of locality and translation equivariance of image features.

ViT as a transformer architecture with global self-attention does not explicitly introduce such properties that are specifically designed for images (remember, the attention mechanism was originally proposed for text data). The primary component of ViT introducing inductive bias are the position embeddings allowing the model to leverage (but not enforcing) the concept of position and locality.

Swin transformer can be localized between ResNet and ViT w.r.t. the inductive bias for images. On the one hand, it does not use convolutional layers but attention layers, leading to less inductive bias than CNNs such as ResNet. On the other hand, it does not use global self-attention but attention restricted to certain windows, emphasizing the importance of locality and introducing more inductive bias than ViT.

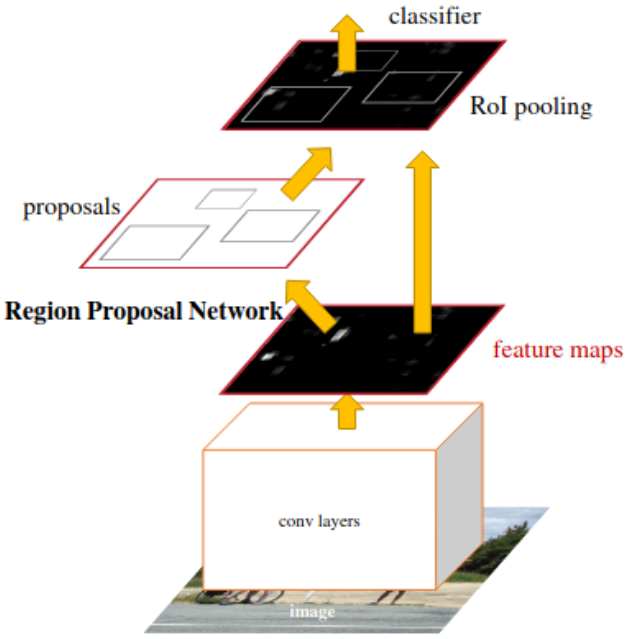

Figure 2: Faster R-CNN architecture. Source: `https://arxiv.org/pdf/1506.01497`.

Exercise 7-2 Object Detection

(a) Why is Faster R-CNN called a two-stage architecture (in comparison to YOLO)?

Detailed Solution

In Faster R-CNN, the so-called Region Proposal Network produces initial bounding box estimates (proposals), which are further refined in a separate prediction head. Therefore, the final bounding box predictions are generated in two stages. One-stage architectures such as YOLO do not have such a refinement step and produce the final box predictions with a single predicition head.

(b) Explain the purpose of RoI pooling in Faster R-CNN.

Detailed Solution

After the region proposal network has generated candidates for bounding boxes, we want to use the image features from these regions to make the final predictions in the detection head. To obtain features of a fixed size (7x7), irrespective of the size and shape of the bounding box proposal, region of interest (RoI) pooling is performed. This is a variant of max-pooling with adaptive window sizes guaranteeing a fixed output size. Additionally, the pooling operation is implemented in a way that is differentiable w.r.t. the proposal box coordinates predicted by the region proposal network.

(c) What are anchor boxes and what is their purpose?

Detailed Solution

Anchor boxes are predefined bounding boxes of specific heights and widths. In object detectors, bounding box prediction subheads are then applied to a specific anchor box shape at every position in the image and predict only the offsets of the actual boxes w.r.t. their anchor. That way, the prediction subheads can specialize for objects of a specific shape and scale, improving the detection performance. As a user, we have to make sure that the object shapes in our dataset are well represented by the anchor boxes. Note that there are also anchor-free object detectors such as DETR. For more information, see here

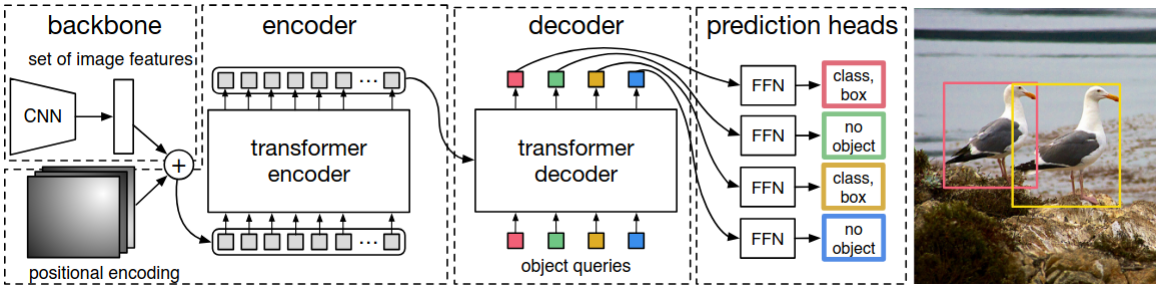

Figure 3: DETR architecture. Source: `https://arxiv.org/pdf/2005.12872`.

(d) What is Non-Maximum Suppression (NMS) in object detection?

Detailed Solution

In object detection, we want to ideally generate exactly one prediction for every object present in the image. However, the sliding window approach of detectors such as Faster R-CNN leads to a lot of predictions, many of which are redundant as they refer to the same object. To alleviate this, NMS is a prediction postprocessing algorithm that suppresses all predictions that are highly overlapped (w.r.t. IoU) by another prediction with a higher confidence (obtained from the classification scores). For more information, see here.

(e) In what regard is the training objective of DETR fundamentally different from preceding object detectors?

Detailed Solution

DETR is a set prediction approach for object detection, i.e., instead of generating a superset of predictions that are then reduced by a postprocessing technique such as NMS, DETR is optimized to directly predict the desired set of objects without redundancies (note that detectors such as Faster R-CNN are not penalized for making two predictions for one object during training). To this end, every prediction has to be made in coordination with other predictions to avoid redundancies. This coordination between predictions is achieved with self-attention layers in the decoder. Consequently, the set of predicted objects has to be matched with ground-truth objects in a one-to-one manner such that exactly one prediction receives positive supervision when multiple or none of the predictions overlap with a ground-truth object. This matching is realized with the Hungarian matching algorithm.

(f) At the example of Mask R-CNN, explain how we can obtain an instance segmentation architecture from an object detection architecture.

Detailed Solution

In principle every object detector can be extended to an instance segmentation model by just adding an additional prediction branch for instance masks next to the class and bounding box predictions (if mask predictions are not made w.r.t. to the corresponding bounding box but the whole image, the mask prediction branch may also replace the bounding box prediction). In Mask R-CNN, the mask prediction branch is simply a sequence of convolution layers operating on the RoI-pooled features.

Exercise 7-3 Segmentation

(a) Explain the differences between semantic segmentation, instance segmentation, and panoptic segmentation.

Detailed Solution

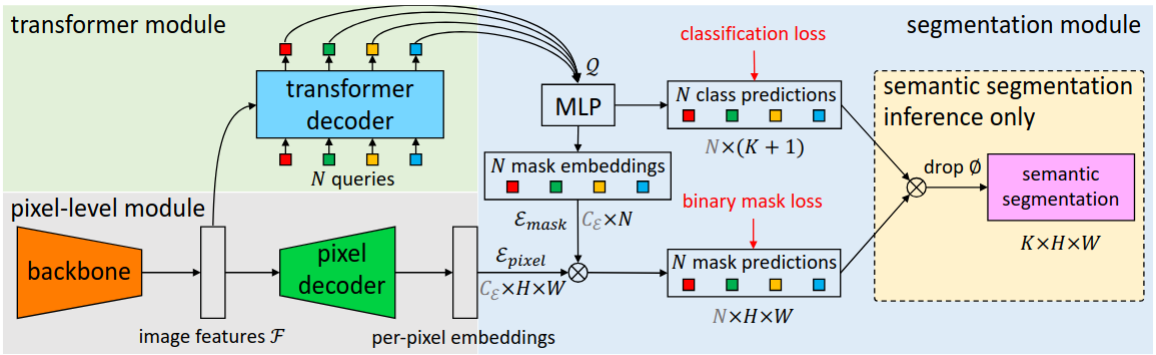

Figure 4: MaskFormer architecture. Source: `https://proceedings.neurips.cc/paper_files/paper/2021/file/950a4152c2b4aa3ad78bdd6b366cc179-Paper.pdf`.

- **Semantic Segmentation:** Every pixel is assigned to exactly one class (classification on pixel level, object instances not distinguished).
- **Instance Segmentation:** Object detection with additional pixel-wise mask predictions (object instances distinguished, no predictions for non-object background pixels).
- **Panoptic Segmentation:** Combination of Semantic and Instance Segmentation. Every pixel is assigned to exactly one class and one instance ID. For pixels assigned to classes representing countable objects/instances (so-called "thing" classes, e.g., person, car), it is enforced that the instance IDs are identical if and only if, the pixels belong to the same instance. For classes representing uncountable concepts (e.g., sky, water), the instance ID is not considered.

(b) Why is MaskFormer fundamentally different from preceding architectures for semantic segmentation?

Detailed Solution

Preceding solutions for semantic segmentation have addressed the task as a per-pixel classification problem. MaskFormer follows a different approach by partitioning an in image into a set of segments (masks) and classifying these as a whole.

(c) Compare the architectures of DETR and MaskFormer.

Detailed Solution

DETR and MaskFormer both follow the set prediction principle and, therefore, both use a transformer decoder operating on input queries and image features coming from a backbone. The output of the transformer decoder are a set of query embeddings which are used to generate predictions. Here, MaskFormer additionally employs a pixel decoder yielding feature maps that are used to create mask predictions via pixelwise multiplication with the query embeddings.

(d) What is the central innovation of Mask2Former compared to MaskFormer on task-level?

Detailed Solution

Mask2Former unifies semantic, instance, and panoptic segmentation, i.e., the same architecture can be applied for all three tasks. This is possible as the concept of mask classification introduced in MaskFormer allows to treat entire masks as instances, providing a neat way to distinguish instances (which is not possible in the per-pixel classification formulation without further ado).