DeepRob

[Student] Lecture 16

by Mohammed Guiga, Danny Langan, Pranav Julakanti

Object Tracking, Transformer Architecture

University of Michigan and University of Minnesota

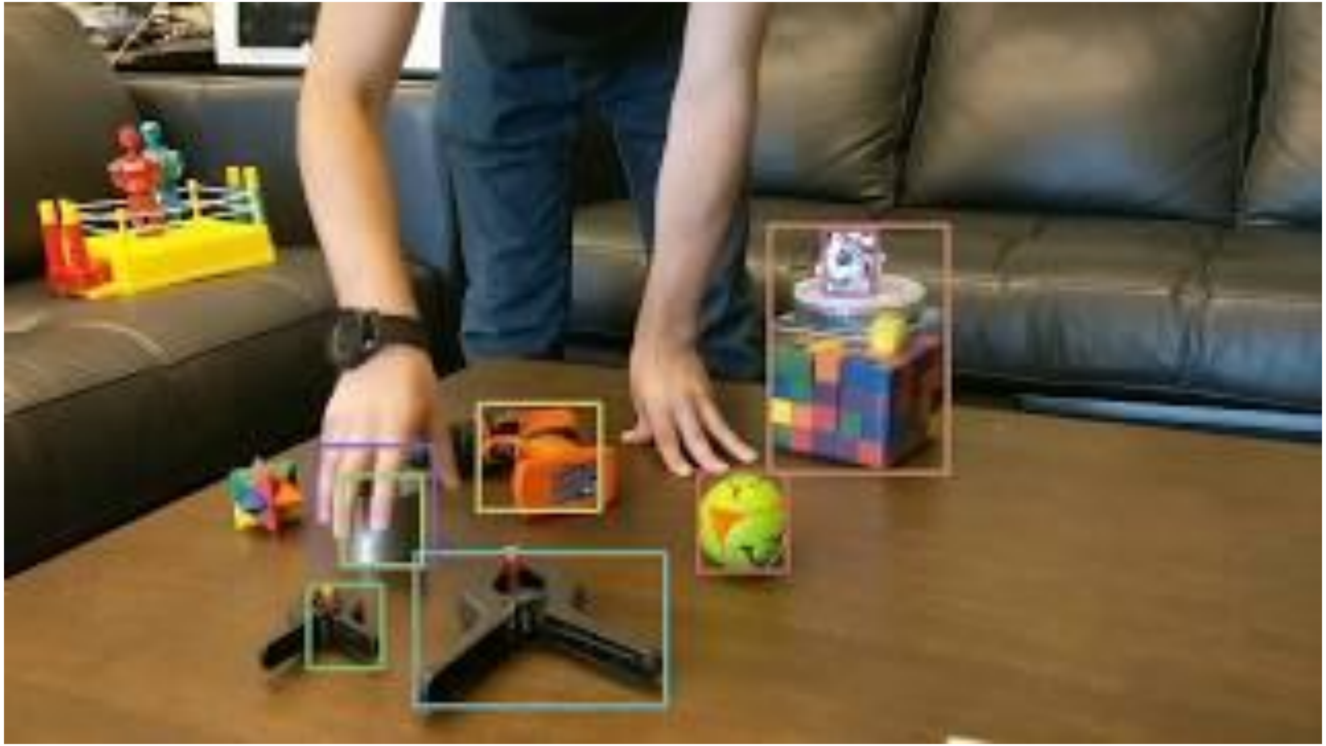
Introduction

- What is tracking?
  - Detecting objects and tracking their movements

- Temporal element in addition to classification

- Example of object tracking before the advent of deep learning:
  - Mean-Shift Tracking
  - Template Matching
  - Optical Flow
  - Kalman Filtering
  - Particle Filtering

Introduction



https://danielgordon10.github.io/papers/re3.html

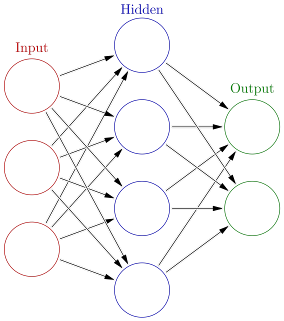
Recurrent Neural Networks (RNN)

- What is a Recurrent Neural Network?
  - A special type of artificial neural network adapted to work for time series data or data involving sequences
- Need to incorporate dependencies between data points
  - RNNs consider the context (hidden state) of previous time steps

Recurrent Neural Networks

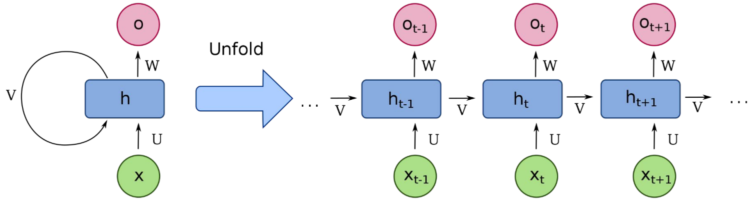
- Feed-forward vs Recurrent neural networks
- Main difference is how the input data is taken in by the model

Feed-forward Neural Network



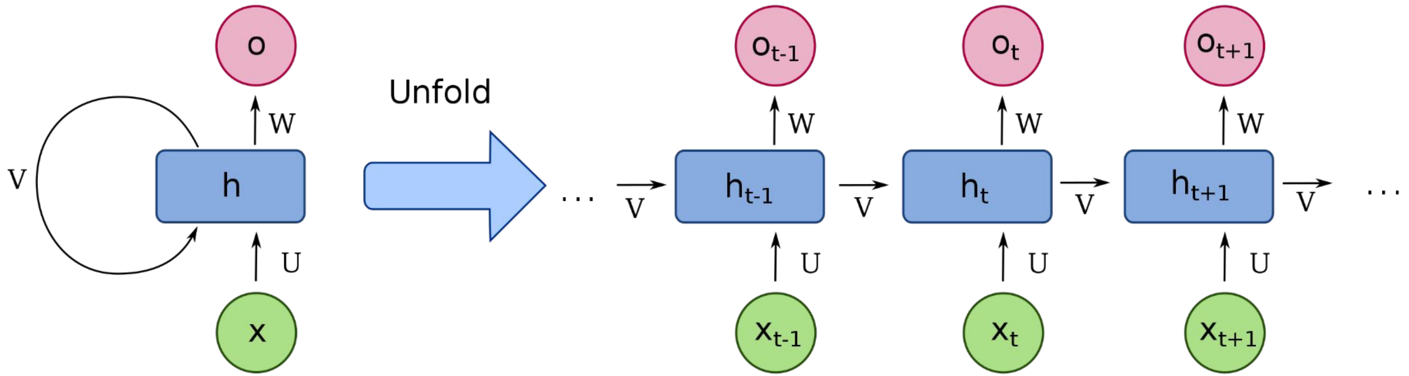
By Glosser.ca - Own work, Derivative of File:Artificial neural network.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24913461

Recurrent Neural Network

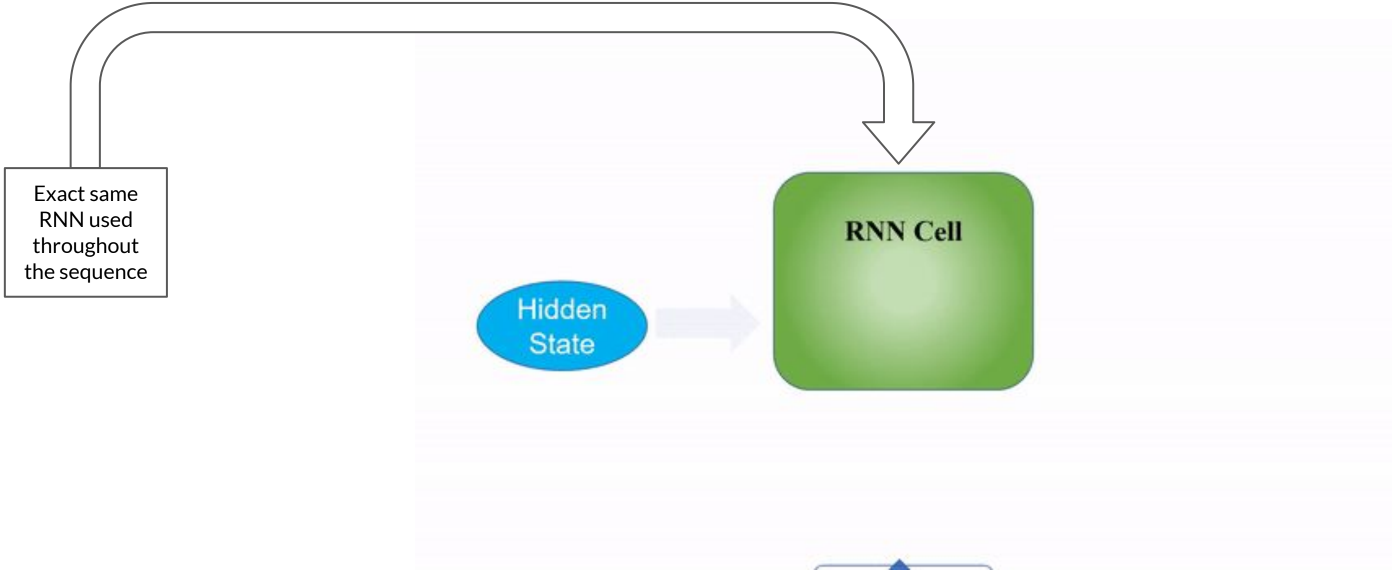


By fdeloche - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60109157

Recurrent Neural Networks



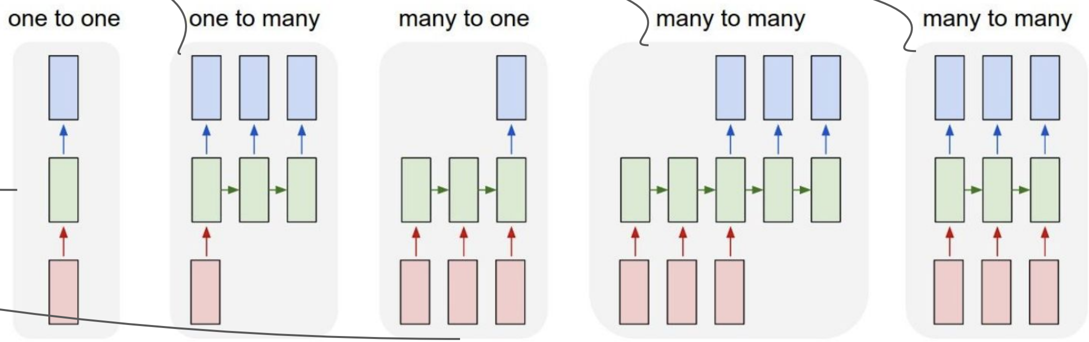
Recurrent Neural Networks



https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

Recurrent Neural Networks

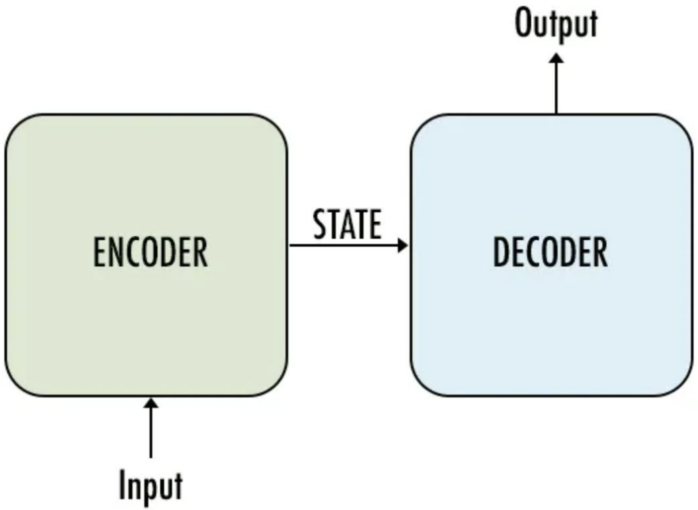
- Traditional feed-forward NN: fixed input -> fixed output
- RNN: (1-N) inputs -> (1-N) outputs
- Classification?
  - One output at the end
- Text generation?
  - An output at each time step



https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

Recurrent Neural Networks

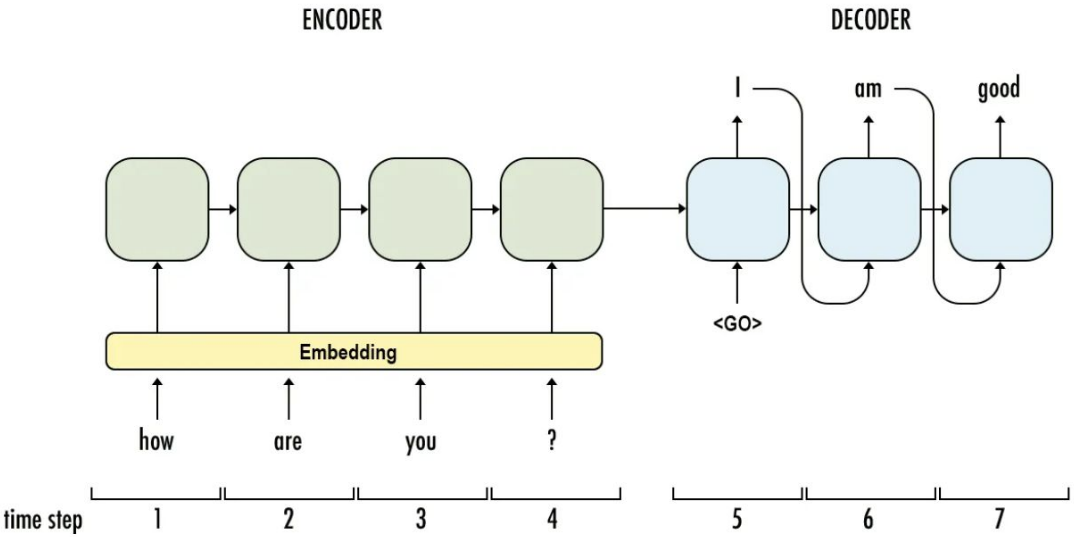
- Sequence to Sequence models (seq2seq)



Source: https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d

Recurrent Neural Networks

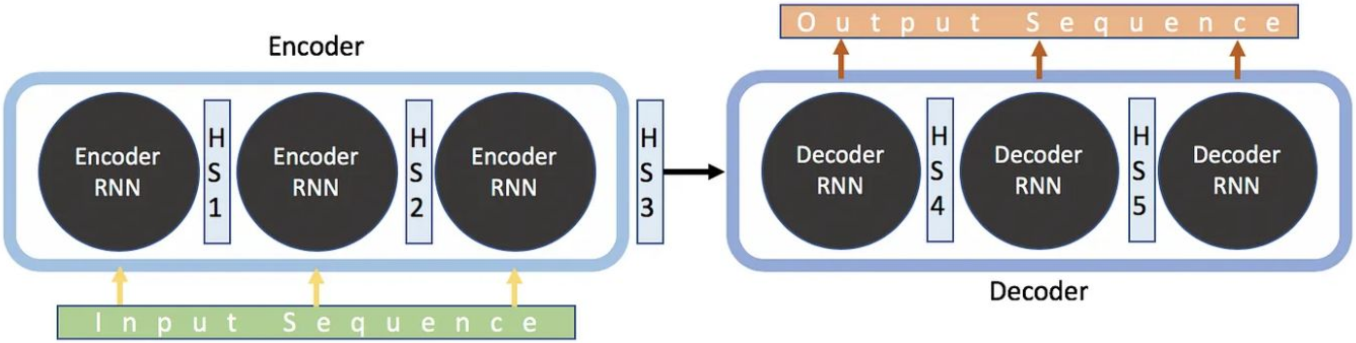
- Sequence to Sequence models (seq2seq)



Source: https://towardsdatascience.com/sequence-to-sequence-model-introduction-and-concepts-44d9b41cd42d

Recurrent Neural Networks

- Sequence to Sequence models (seq2seq)



Source: https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

Recurrent Neural Networks

Initialize hidden state as a matrix of zeros

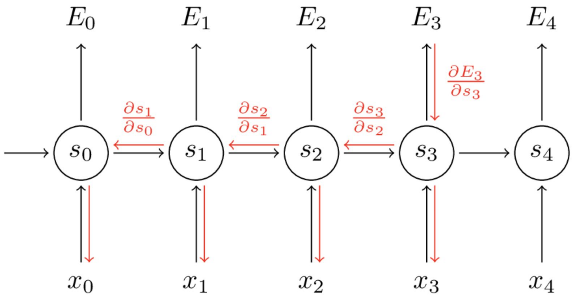
$$\text{hidden}_t = \text{F}(\text{hidden}_{t-1}, \text{input}_t)$$

$$\text{hidden}_t = \tanh(\text{weight}_{hidden} * \text{hidden}_{t-1} + \text{weight}_{input} * \text{input}_t)$$

If output:

$$\text{output}_t = \text{weight}_{output} * \text{hidden}_t$$

Training and Backpropagation



https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/

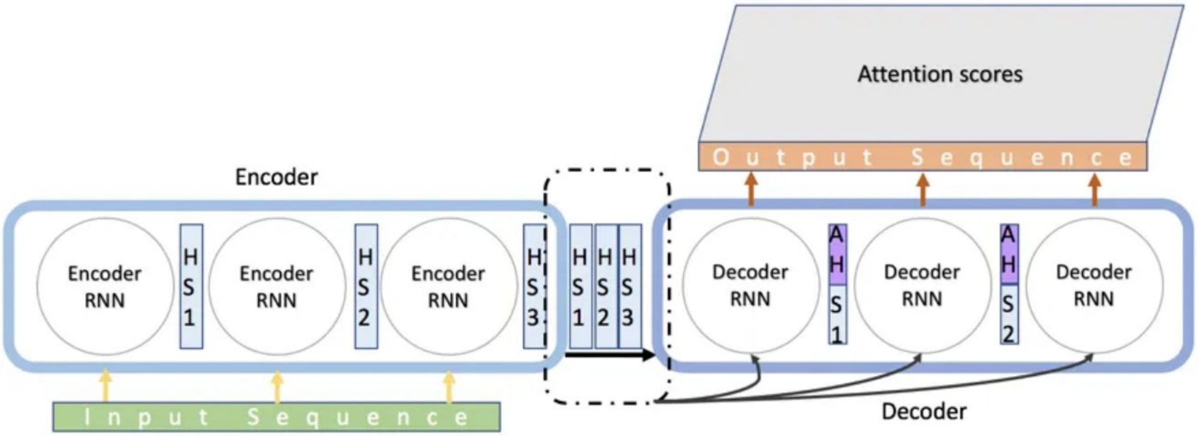
All the weights are exactly the same - weights of the networks are shared temporally

Recurrent Neural Networks (RNN)

- RNNs allow us to carry information through time - Cool!
- But what are the downsides?
- Vanishing / exploding gradients
- Arises during back propagation
- Continuous matrix multiplications can cause the gradients to shrink (vanish) or inflate (explode)

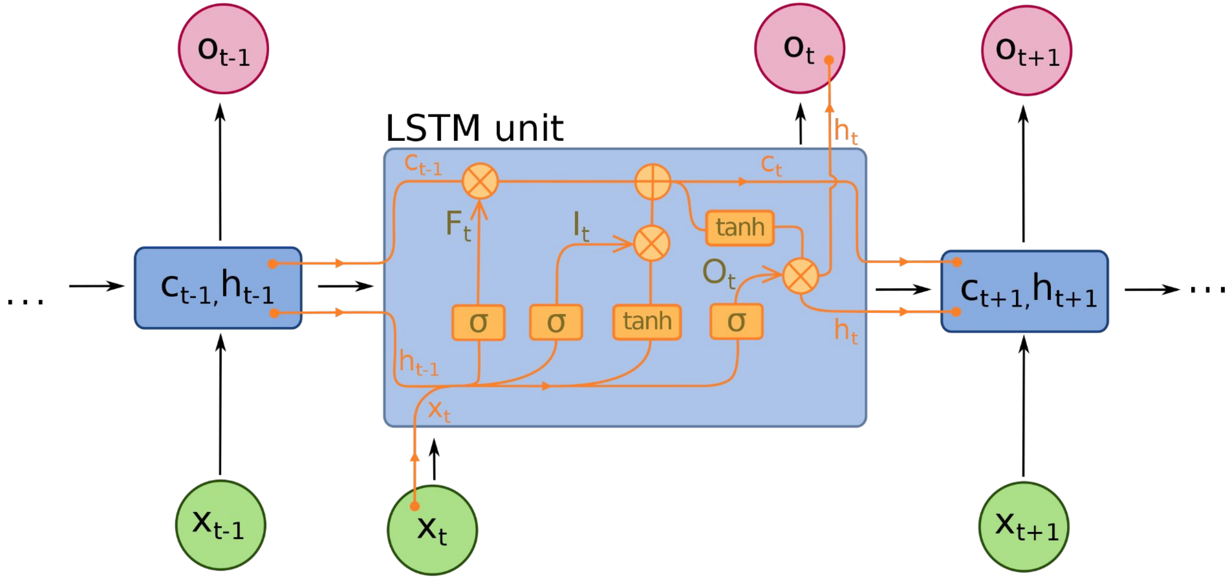
Recurrent Neural Networks

- Sequence to Sequence models (seq2seq)



Source: https://towardsdatascience.com/day-1-2-attention-seq2seq-models-65df3f49e263

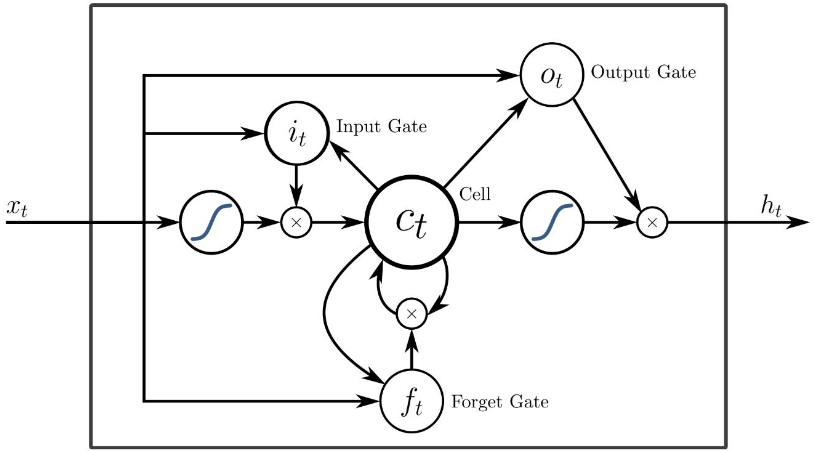
Long Short-Term Memory (LSTM)



By fdeloche - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=60149410

Long Short-Term Memory (LSTM)

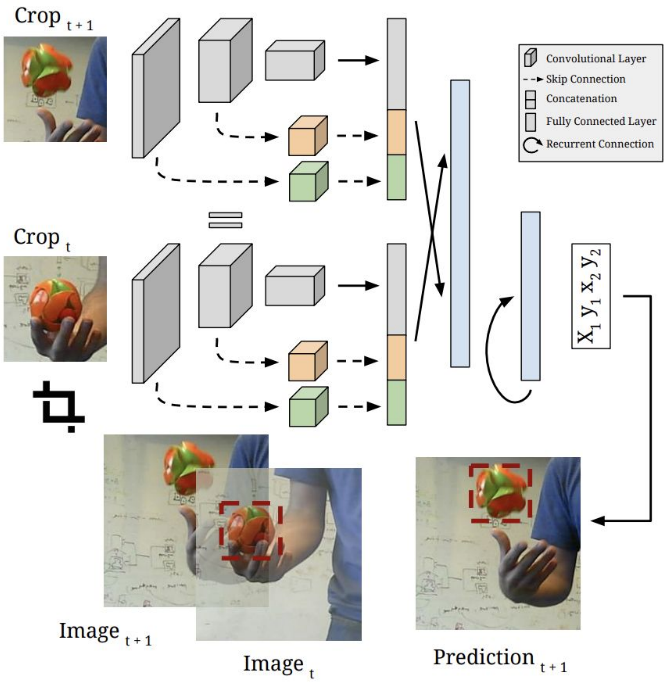
- **Input gate:** regulates the input into the unit/layer
- **Output gate:** regulates the output from the unit
- **Forget gate:** regulates what the cell should forget



https://ai.stackexchange.com/questions/18198/what-is-the-difference-between-lstm-and-rnn

RNNs and Tracking

- Image crop pairs fed in at each timestep
- Add a skip layer before each pooling stage
    - This is to preserve high-resolution spatial information
- Weights from two images are shared
- Output from convolutional layers fed into a fully-connected layer and LSTM



https://danielgordon10.github.io/pdfs/re3.pdf

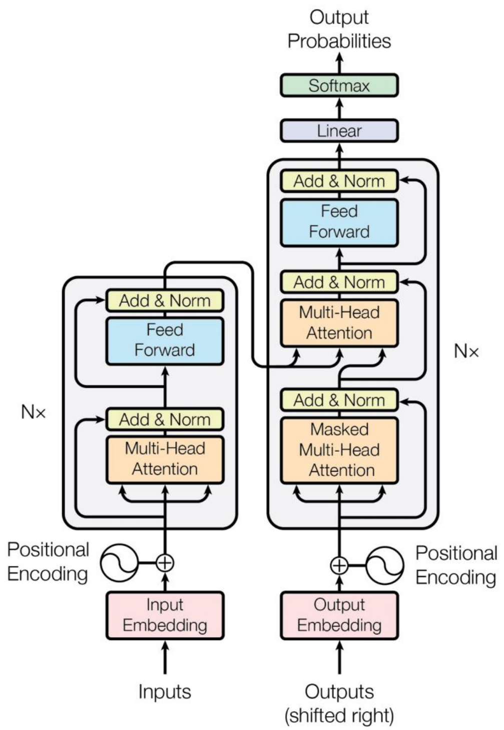
Transformers

Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Transformers

Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

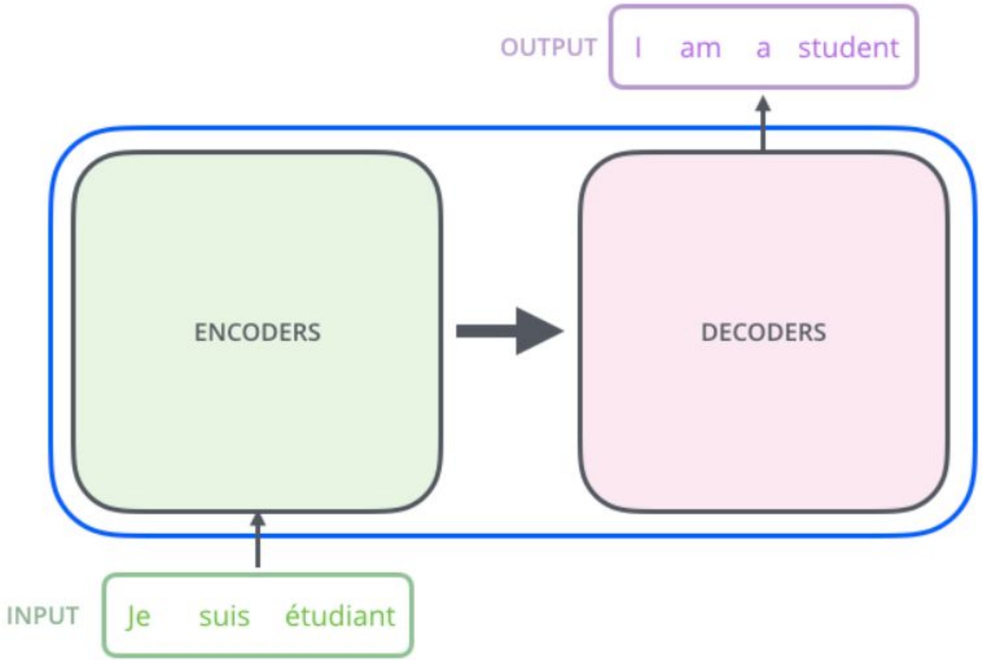
Transformer Progression



Transformer
Vaswani et al, 2017

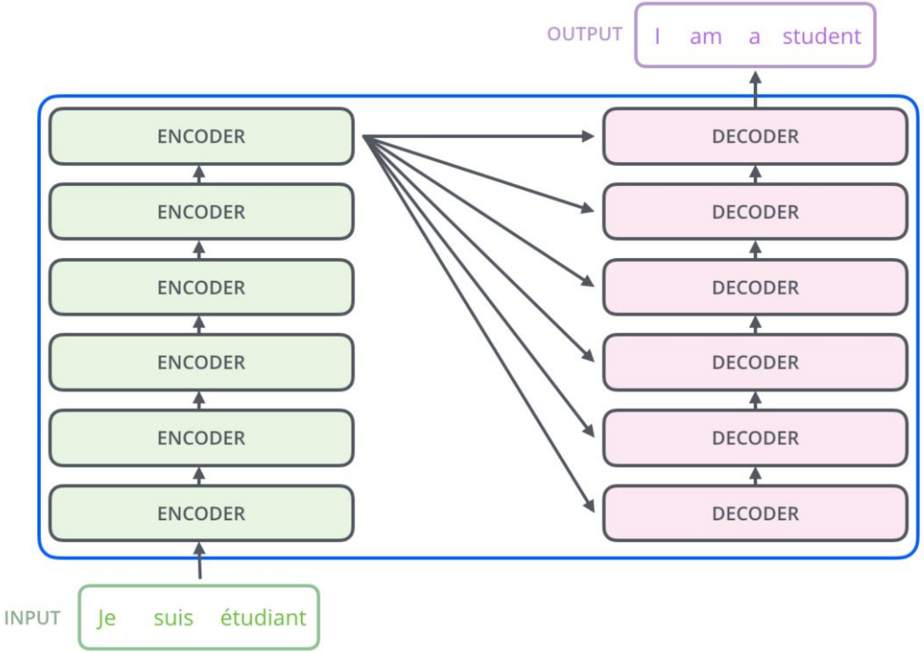
Mohit Shridhar, Acting with Perception and Language,
https://rpm-lab.github.io/CSCI5980-Spr23-DeepRob/assets/slides/acting_with_perception_and_language_(mohit_shridhar).pdf

Encoders/Decoders



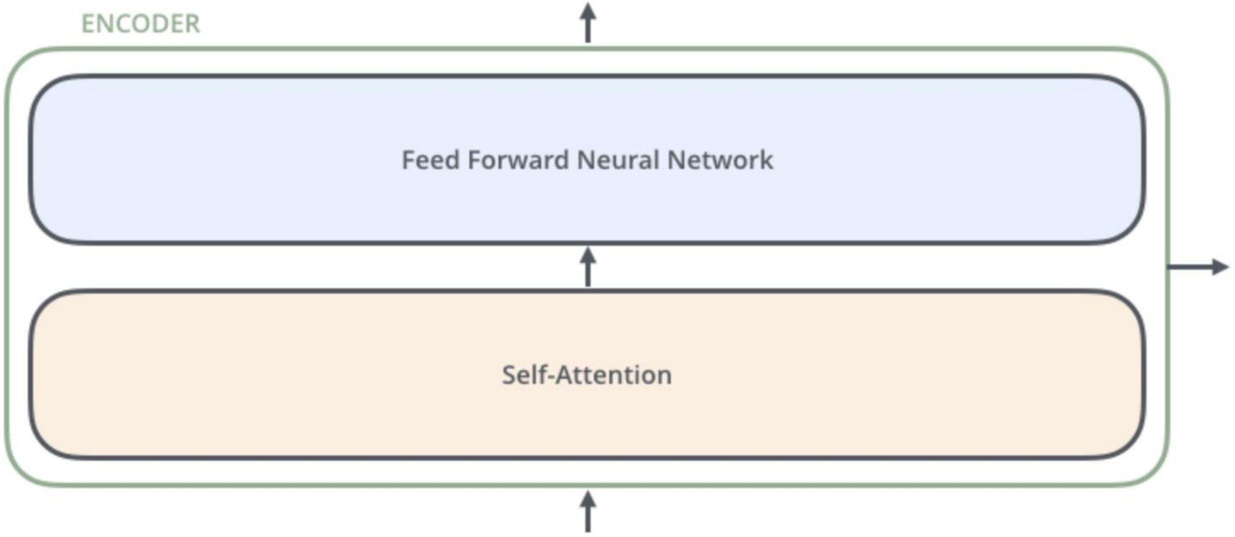
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoders/Decoders



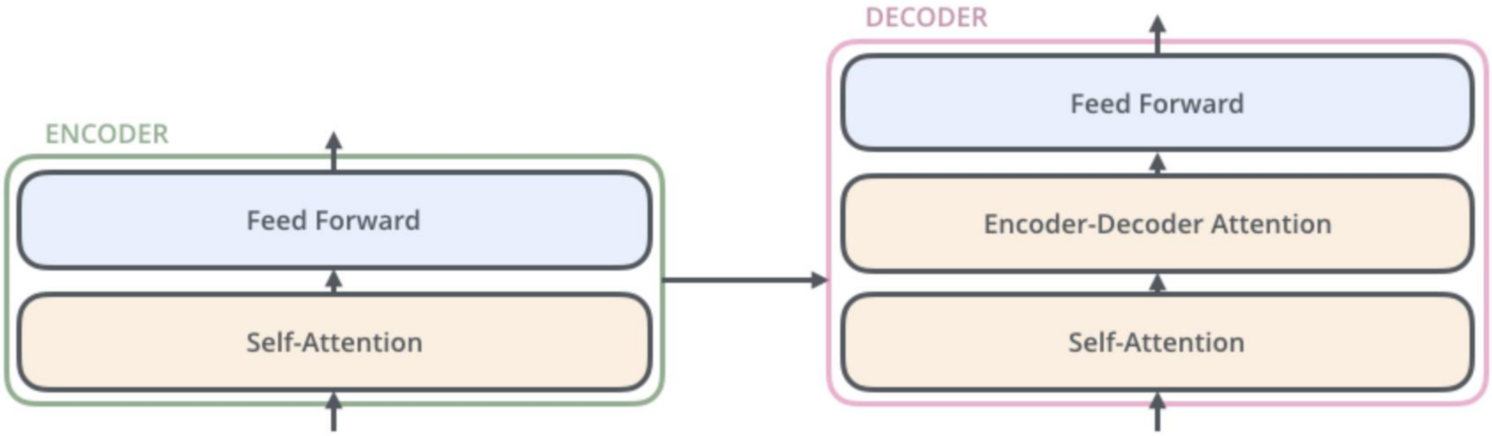
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoder Block Architecture



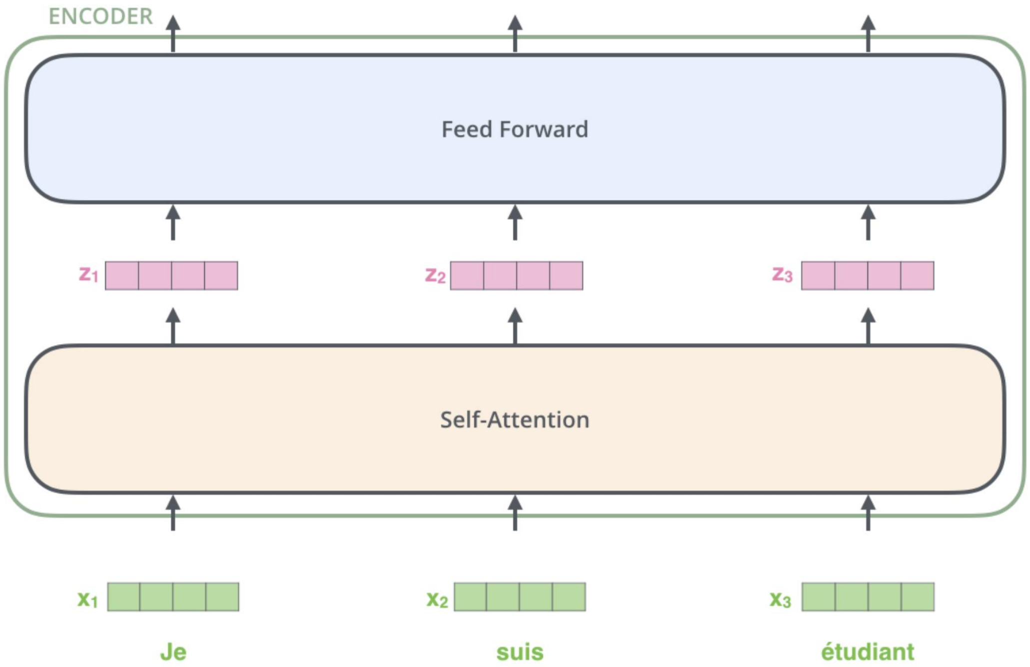
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoder Block Architecture



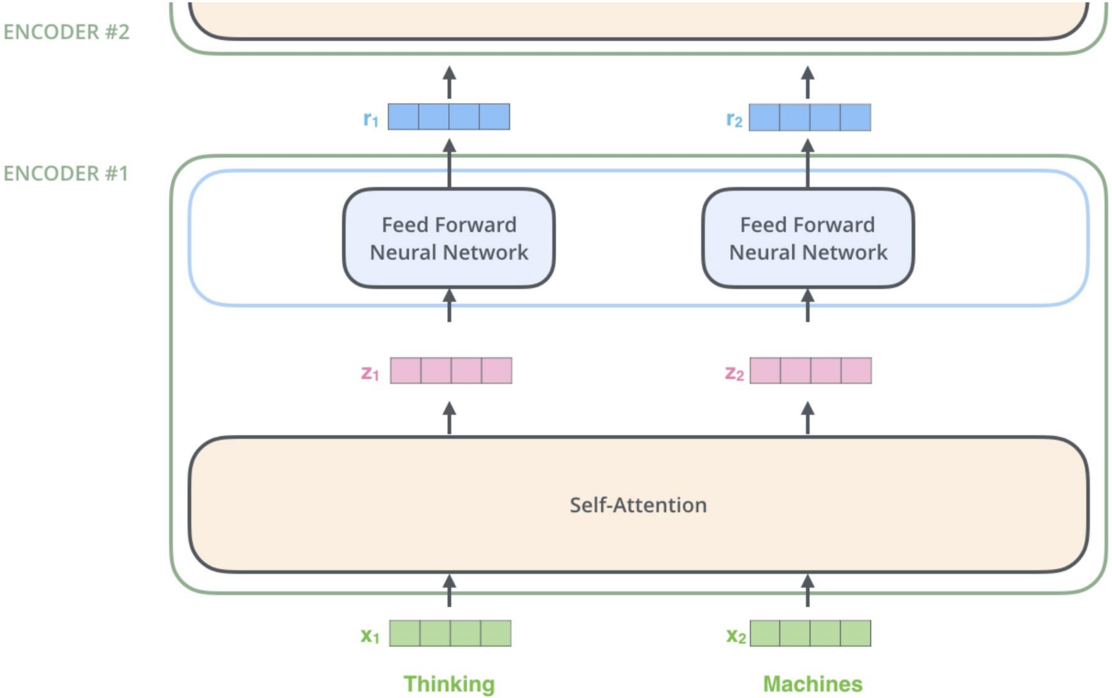
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoder Block Architecture



Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

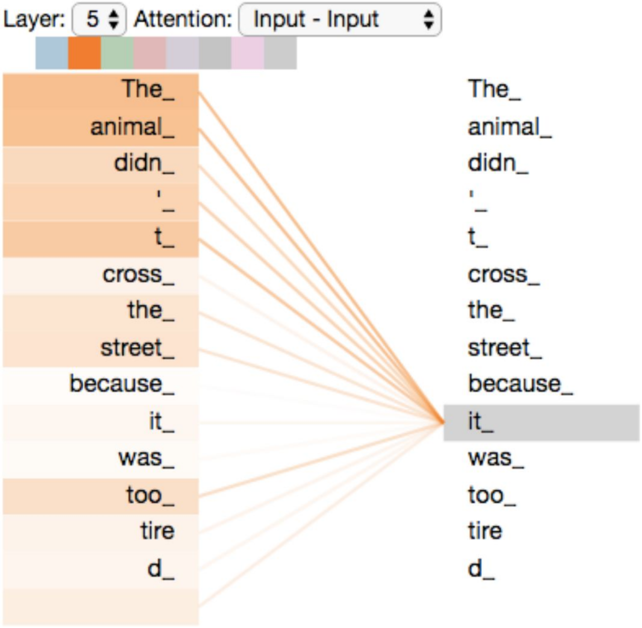
Encoder Block Architecture



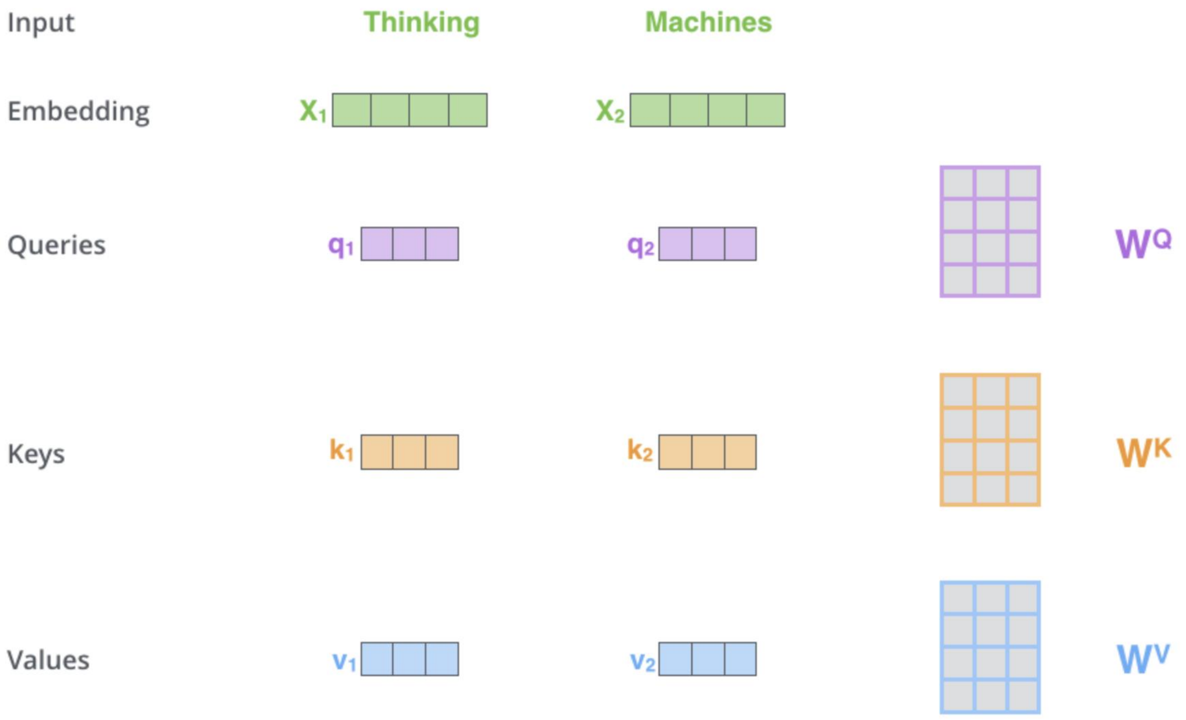
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Self-Attention

"The animal didn't cross the street because it was too tired"

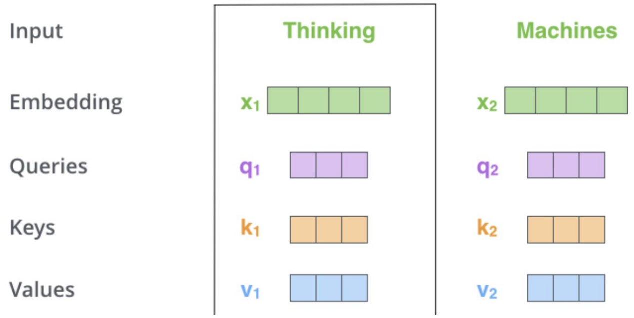


Self-Attention



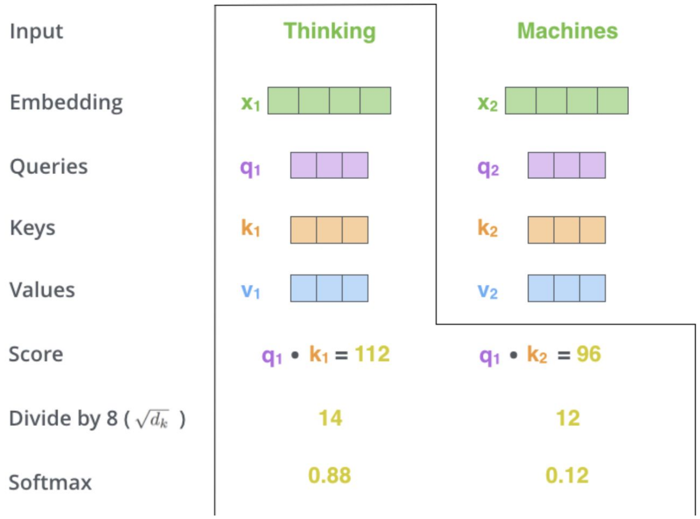
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Self-Attention



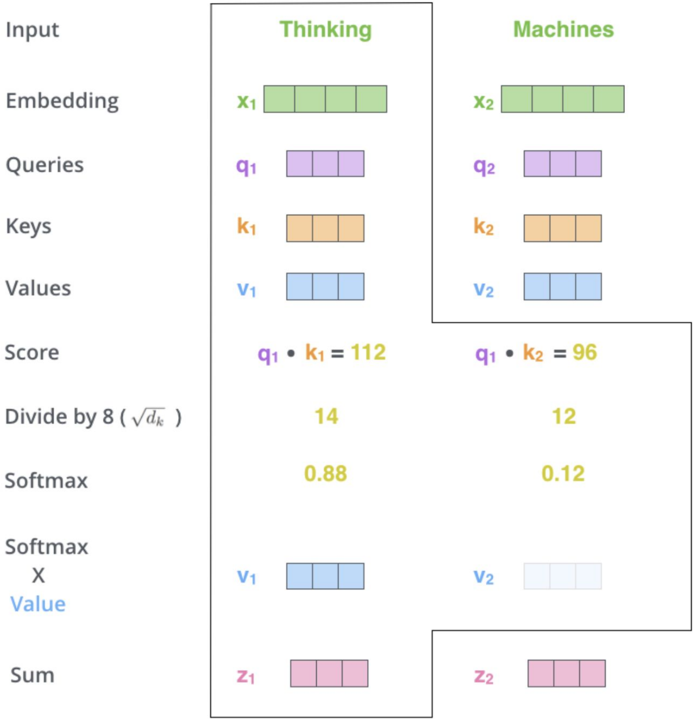
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Self-Attention



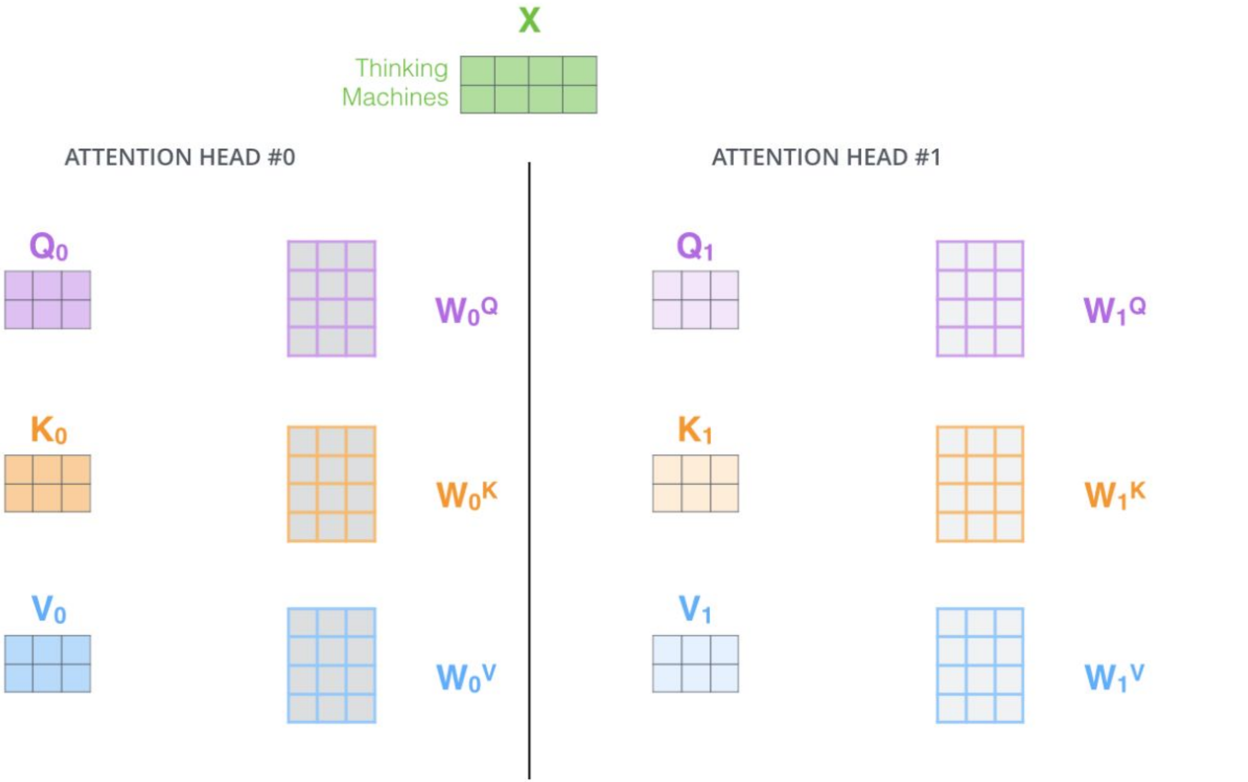
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Self-Attention



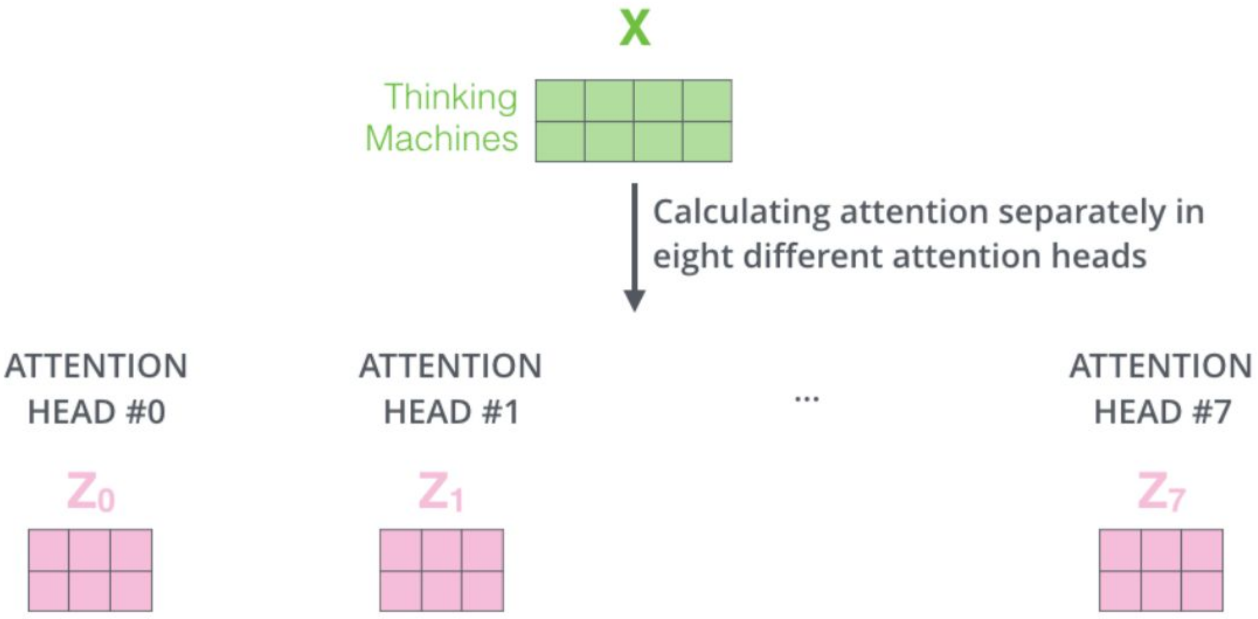
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention



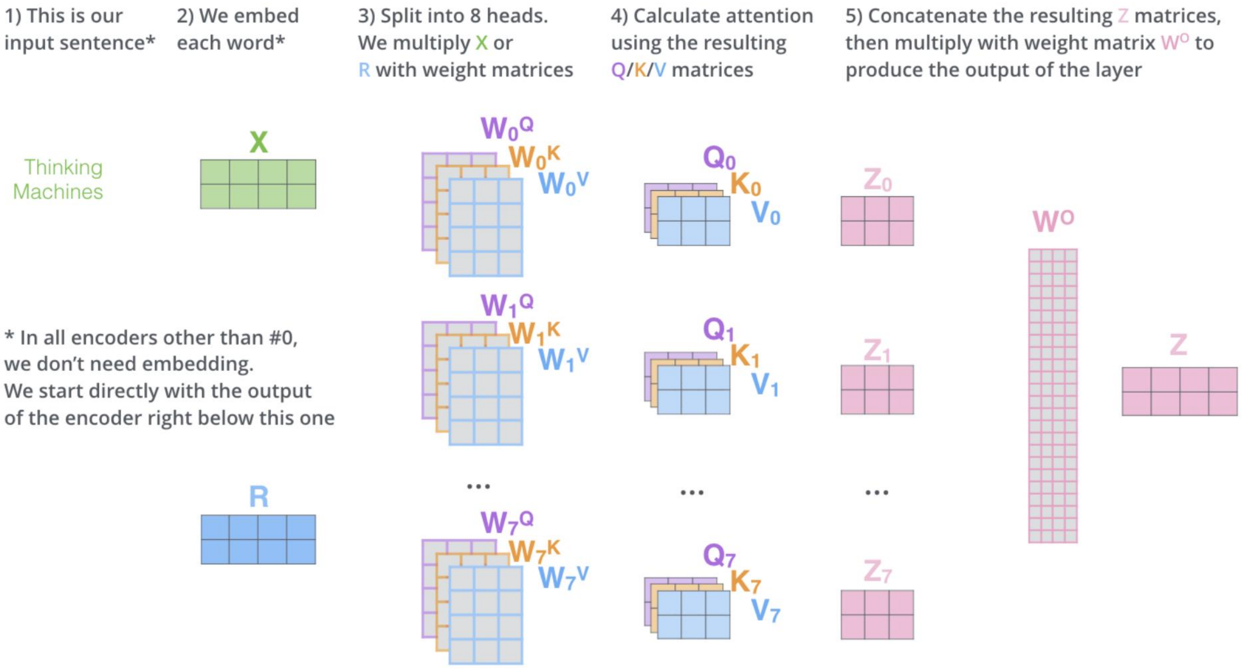
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention



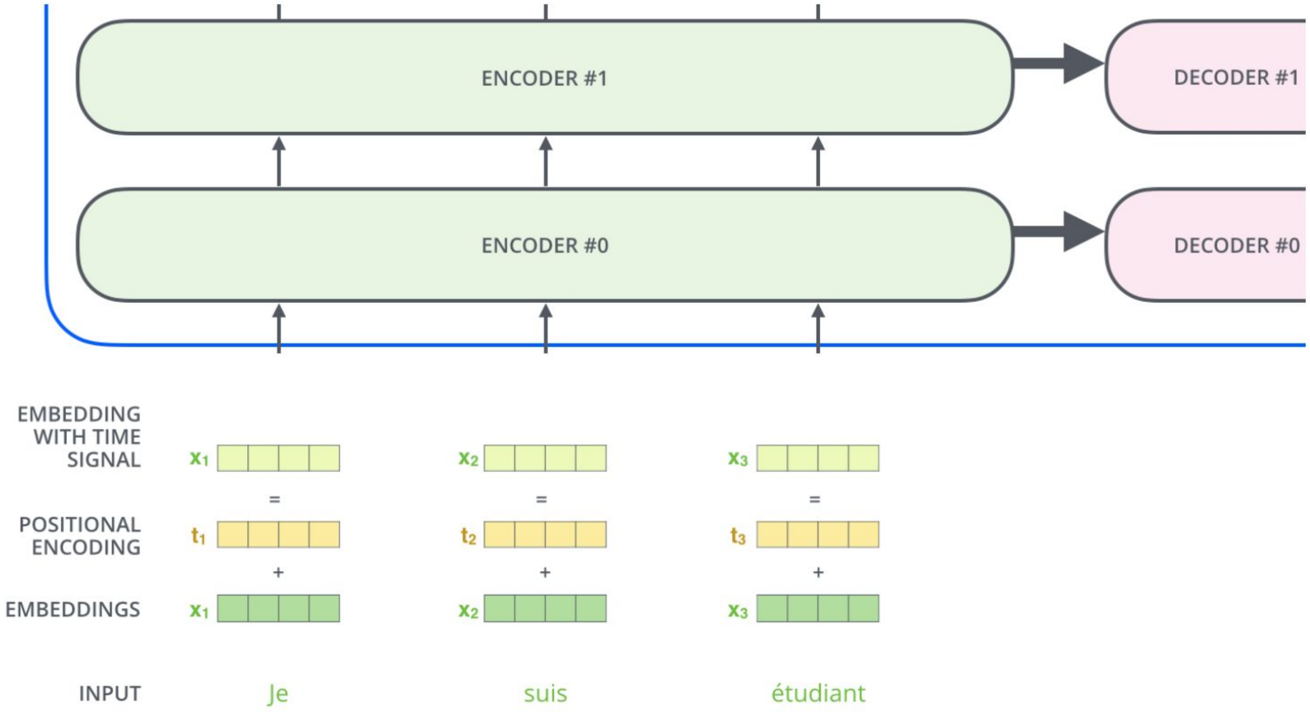
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention



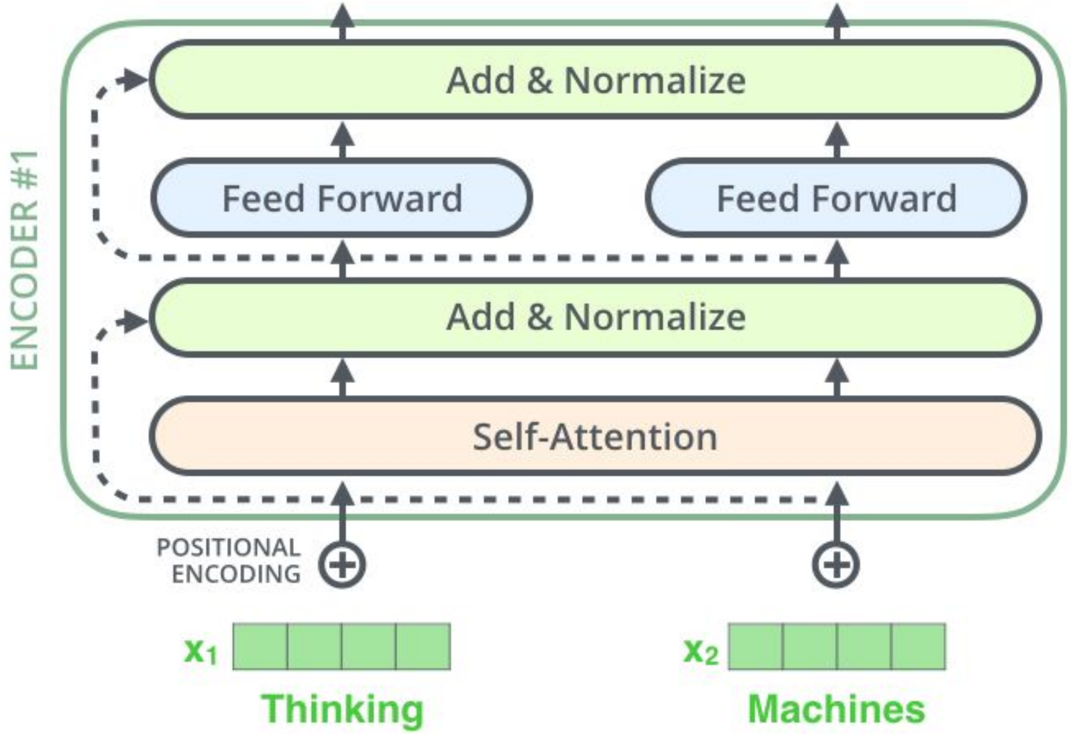
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Embedding Inputs



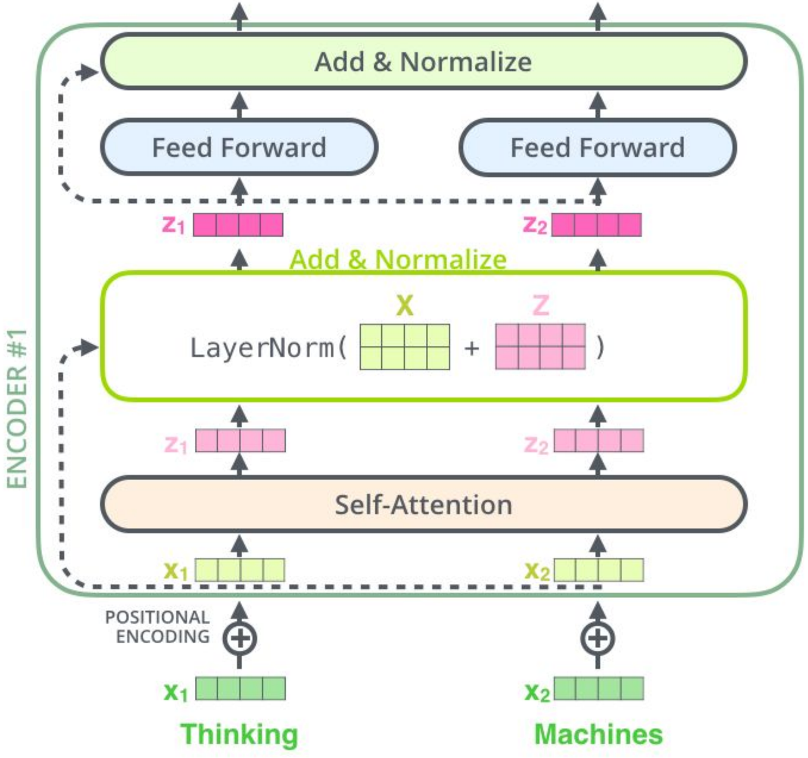
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoder Structure



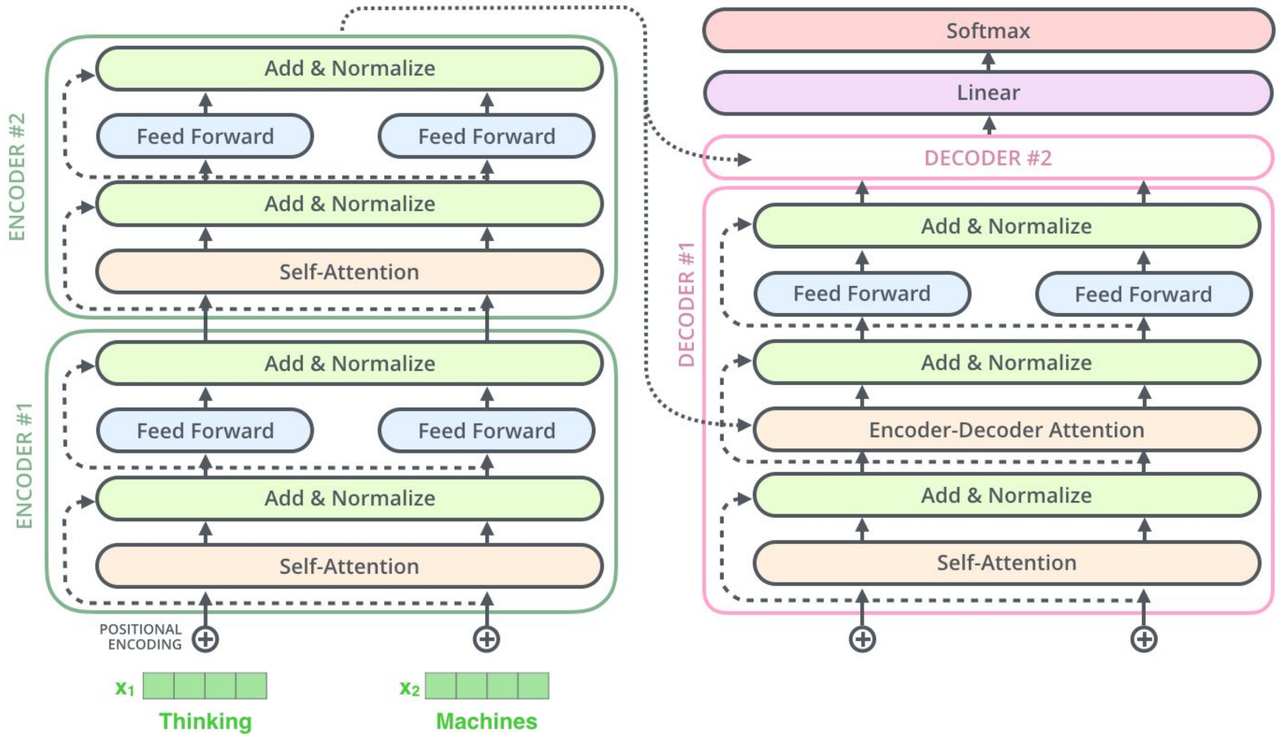
Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoder Structure



Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Encoder/Decoder Structure



Jay Allamar, The Illustrated Transformer, https://jalammar.github.io/illustrated-transformer/

Transformer Progression

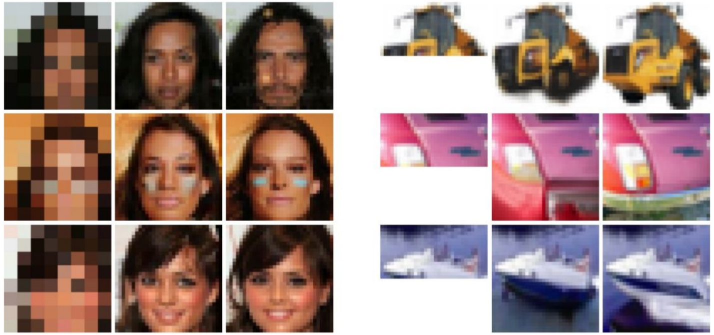


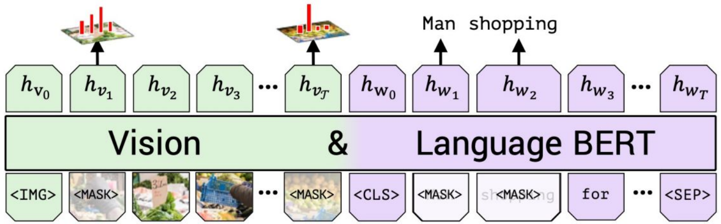
Image Transformer
Parmar et al, 2018

Transformer
Vaswani et al, 2017

Problem:

■ vs. word

Transformer Progression



Vision-Lang BERTs
Lu et al, 2019
Tan et al, 2019

Image Transformer
Parmar et al, 2018

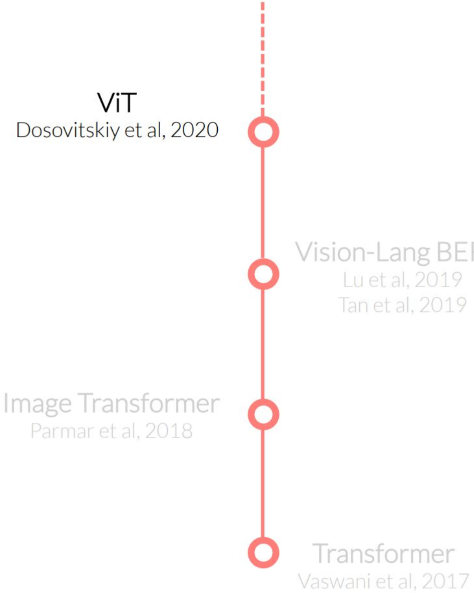
Transformer
Vaswani et al, 2017

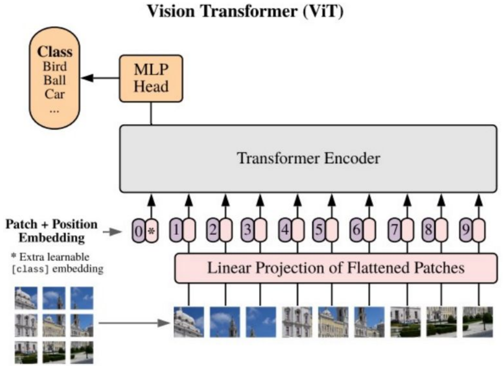
Problem:

Object Detections

Mohit Shridhar, Acting with Perception and Language,
https://rpm-lab.github.io/CSCI5980-Spr23-DeepRob/assets/slides/acting_with_perception_and_language_(mohit_shridhar).pdf

Transformer Progression

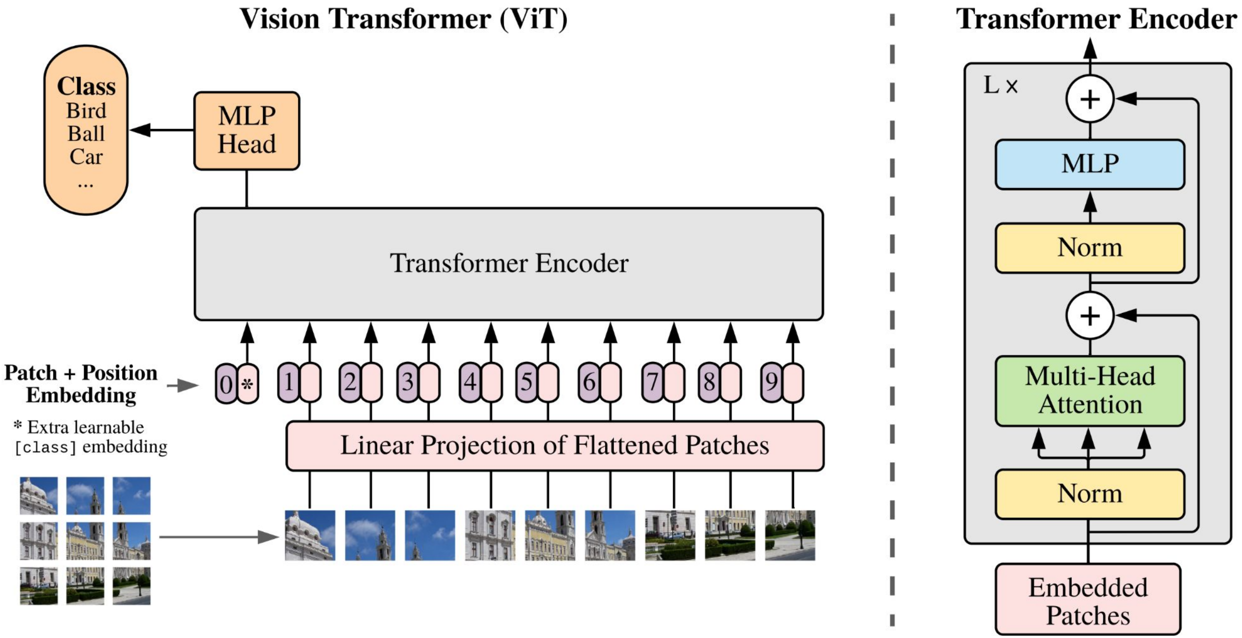




Solution:

2D patches!

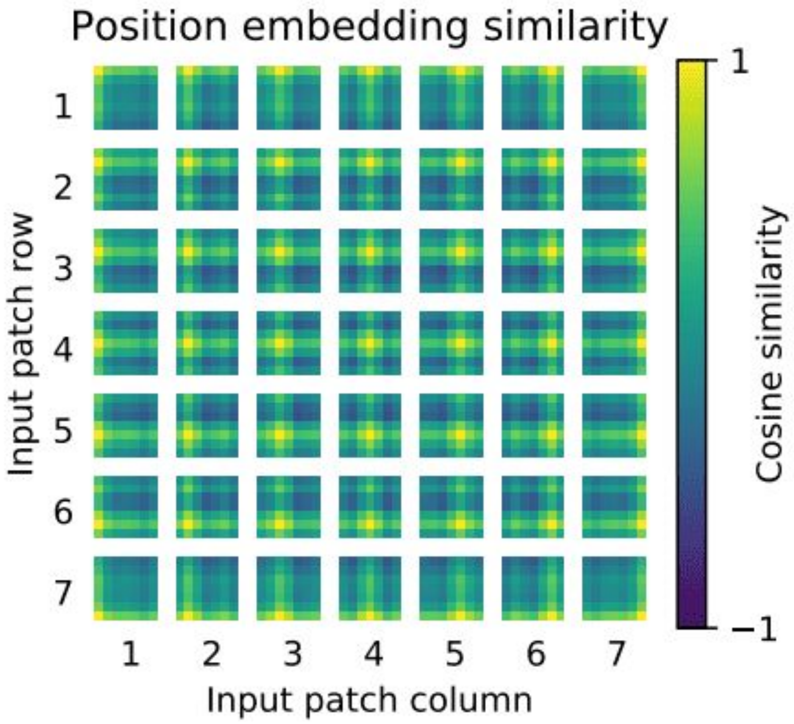
Visual Transformer



Dosovitskiy et al., AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE,
https://arxiv.org/pdf/2010.11929.pdf

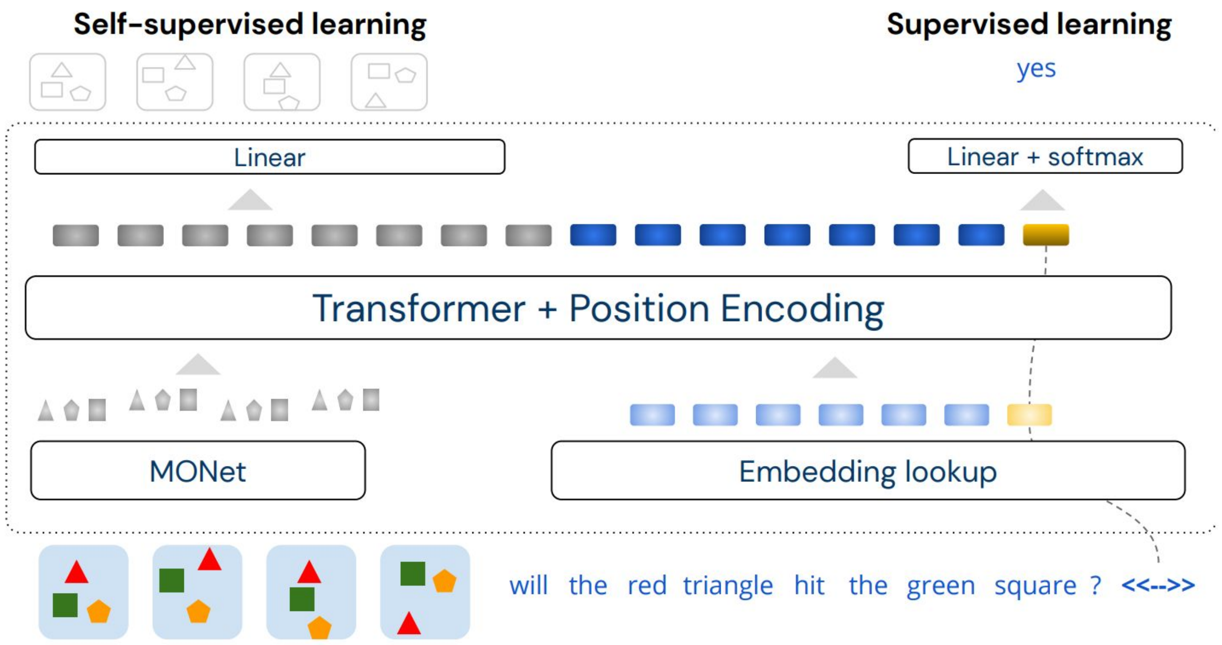
Visual Transformer

- Similarity of position embeddings of ViT-L/32.
- Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches.



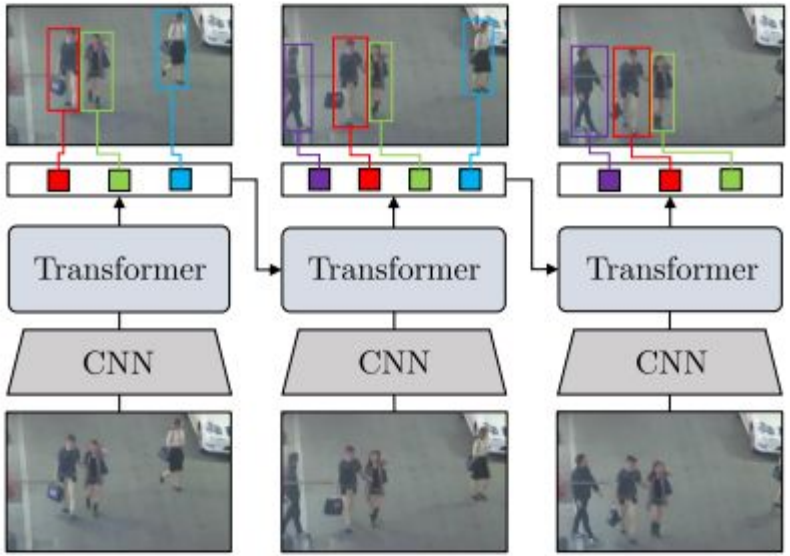
Dosovitskiy et al., AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE, https://arxiv.org/pdf/2010.11929.pdf

ALOE - Attention Over Learned Object Embeddings



Ding et al., Attention over learned object embeddings enables complex visual reasoning, https://arxiv.org/pdf/2012.08508.pdf

Trackformer



T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "TrackFormer: Multi-Object Tracking with Transformers," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, URL: https://arxiv.org/abs/2101.02702.

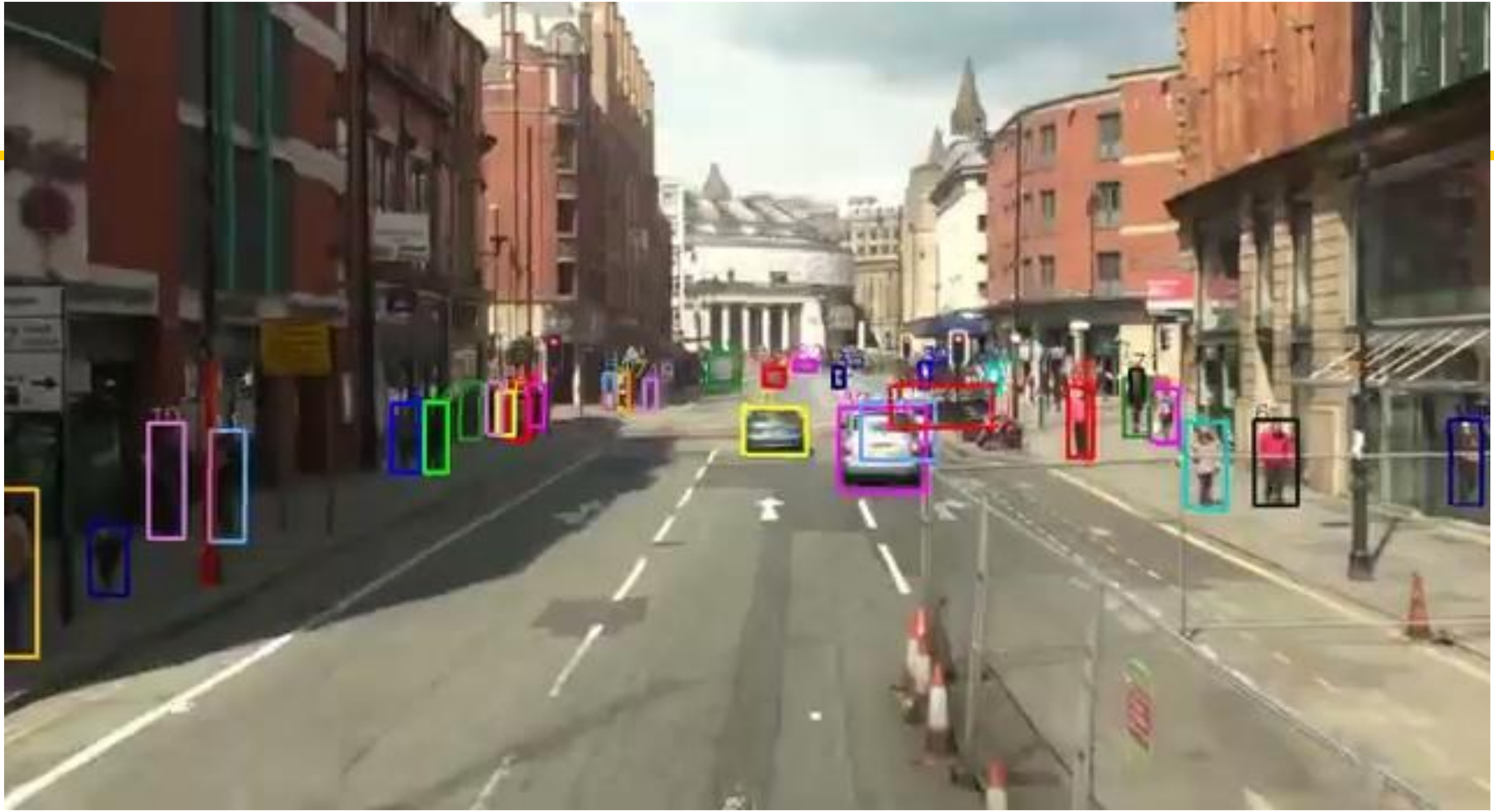
Trackformer Background: Multi Object Tracking (MOT)

- Goal of the paper is to track and discriminate up to *K* distinct individuals over the course of *T* frames
- A *track* is a set of bounding boxes for a single individual over many time steps

$$b_t^k = [x_t^k, y_t^k, w_t^k, h_t^k]$$

$$V = [f_1, \dots, f_T]$$

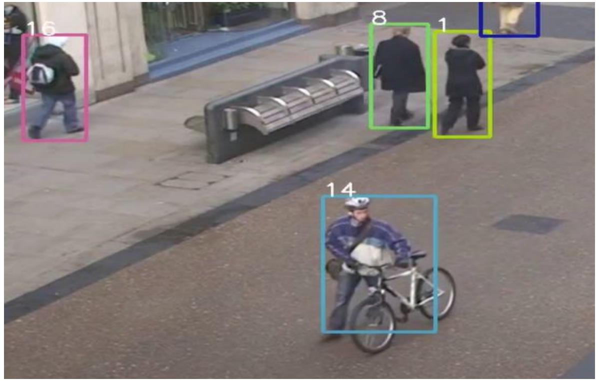
$$T_k = [b_{t_1}^k, \dots, b_{t_n}^k]$$



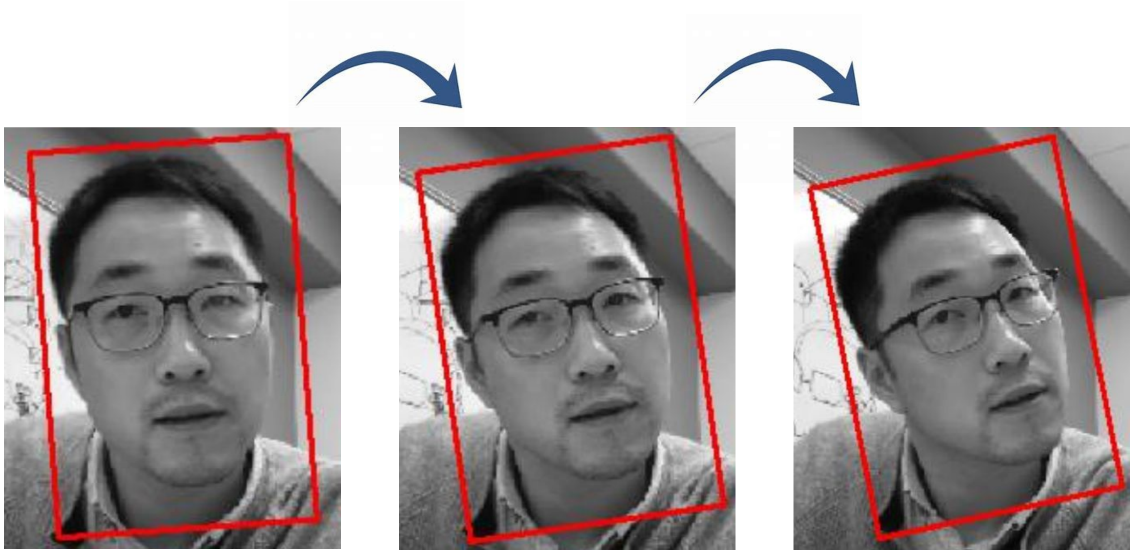
MOT17-13-SDP Ground Truth: [MOT Challenge - Visualize]()

Trackformer Background: Tracking By Detection

- Given a set of detections how do we associate between frames?
- Paper goes over many approaches:
  - Motion based
  - Feature based
  - Cost minimizing objective functions



Trackformer Background: Tracking by regression



1. Compute Jacobian $\frac{\partial W}{\partial p}$

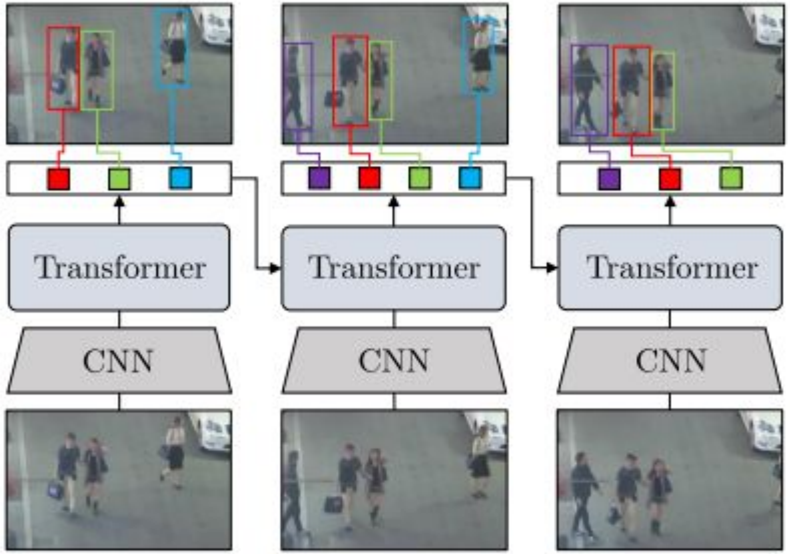
---

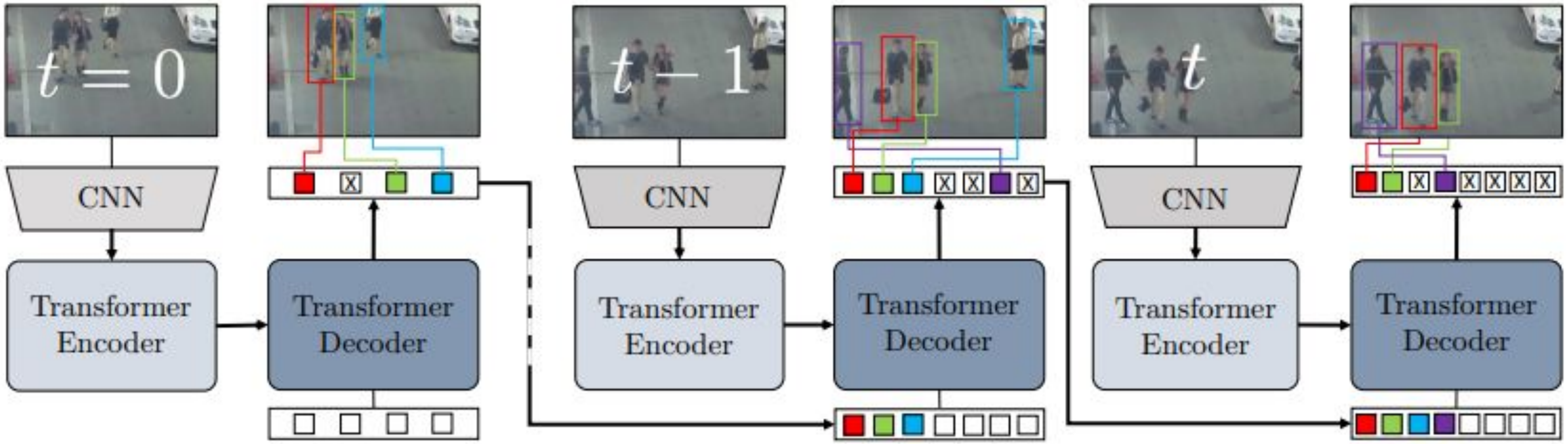
2. Warp the target image $I(W(x;p))$
3. Compute the error image $T(x) - I(W(x;p))$
4. The gradient image $\nabla T(x)$
5. Compute steepest descent images $\nabla T \frac{\partial W}{\partial p}$
6. Compute Hessian $H = \sum_x \left(\nabla T \frac{\partial W}{\partial p} \right)^T \left(\nabla T \frac{\partial W}{\partial p} \right)$
7. Compute $\Delta p = H^{-1} \sum_x \left(\nabla T \frac{\partial W}{\partial p} \right)^T (T(x) - I(W(x;p)))$
8. Update $W(x;p) \leftarrow W(x;p) \circ W^{-1}(x;\Delta p)$
9. Goto 2 unless $\|\Delta p\| < \varepsilon$

Hyun Soo Park Image Alignment Lecture: https://www-users.cse.umn.edu/~hspark/csci5561_S2021/csci5561.html

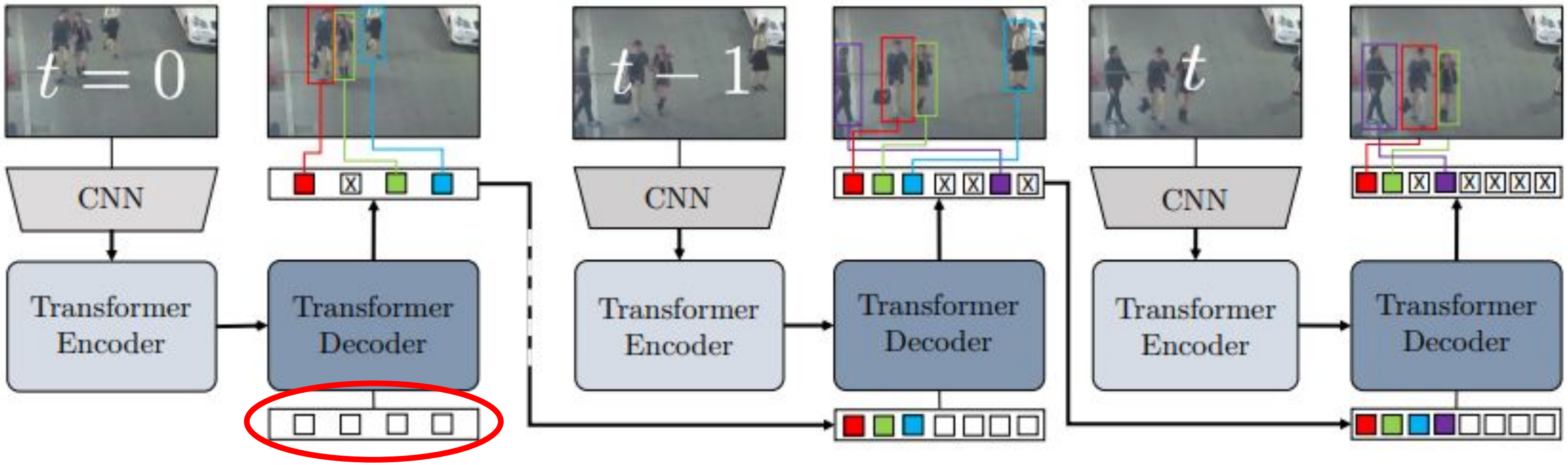
Our Project: Trackformers

- Uses Transformers to do multi-object tracking
- Extends the Transformer concept from linguistic to the visual domain
- Uses the intuition that humans use attention to track objects
- First to use Transformers for both Detection and frame association

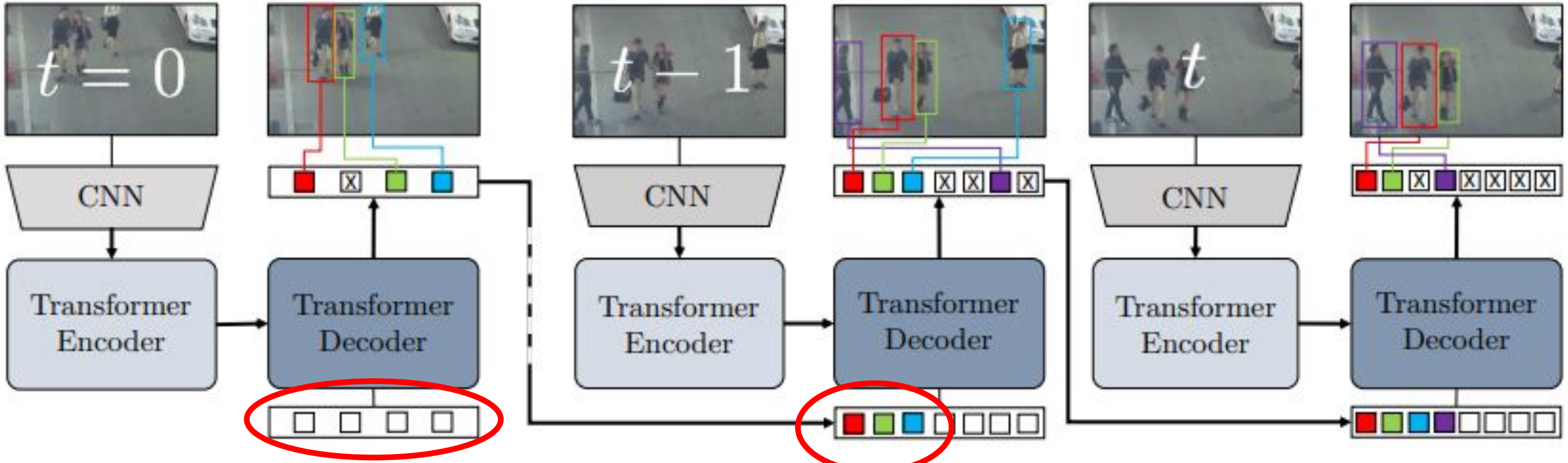




$t = 0$
$t - 1$
t
CNN
Transformer Encoder
Transformer Decoder
CNN
Transformer Encoder
Transformer Decoder
CNN
Transformer Encoder
Transformer Decoder



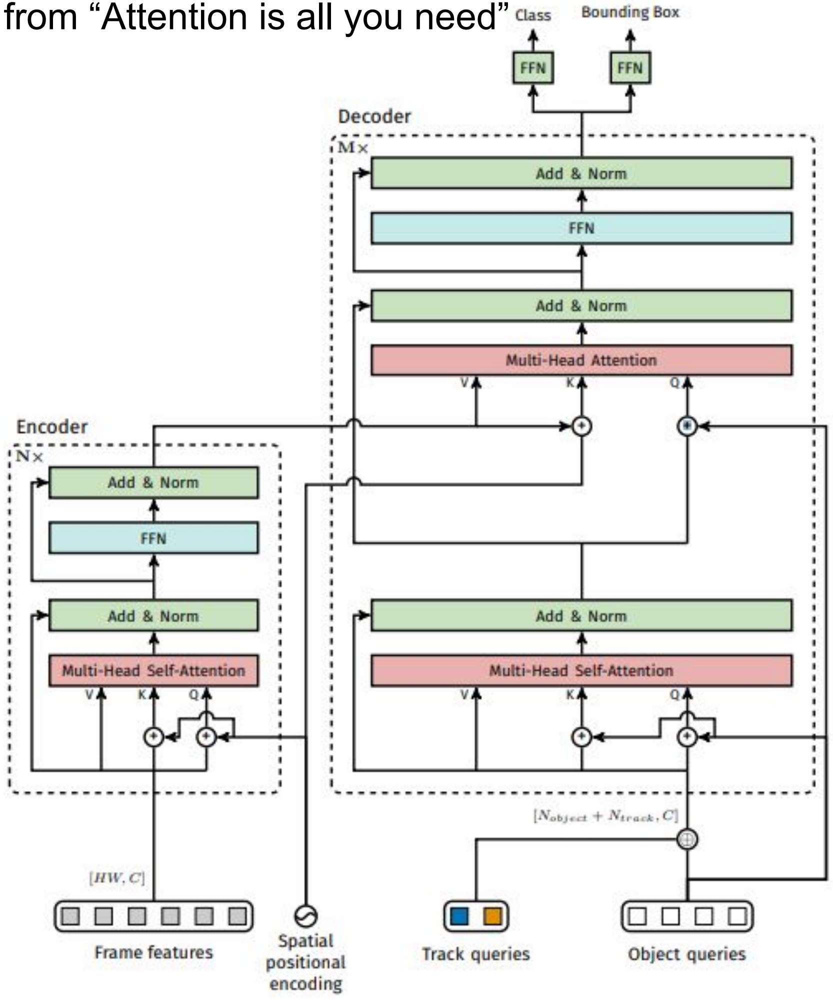
Fixed static
object queries



Fixed static
object queries

Dynamic track queries

Paper Uses Transformer from "Attention is all you need"



Architecture Overview

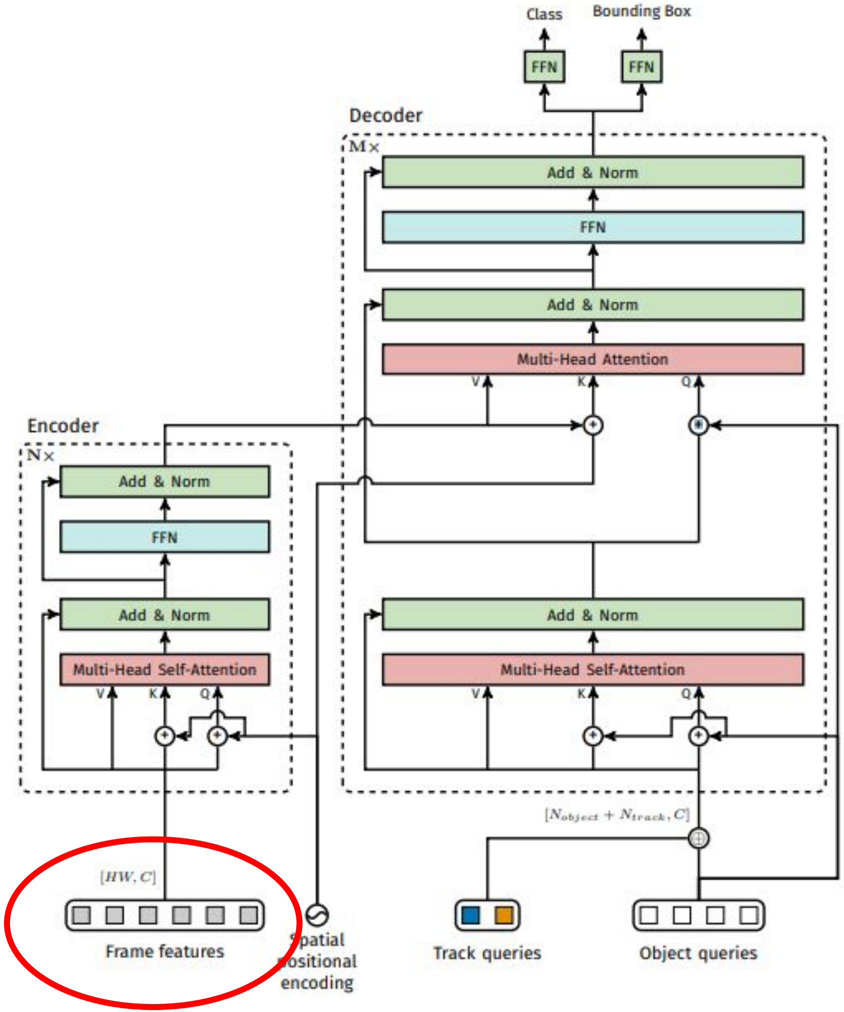
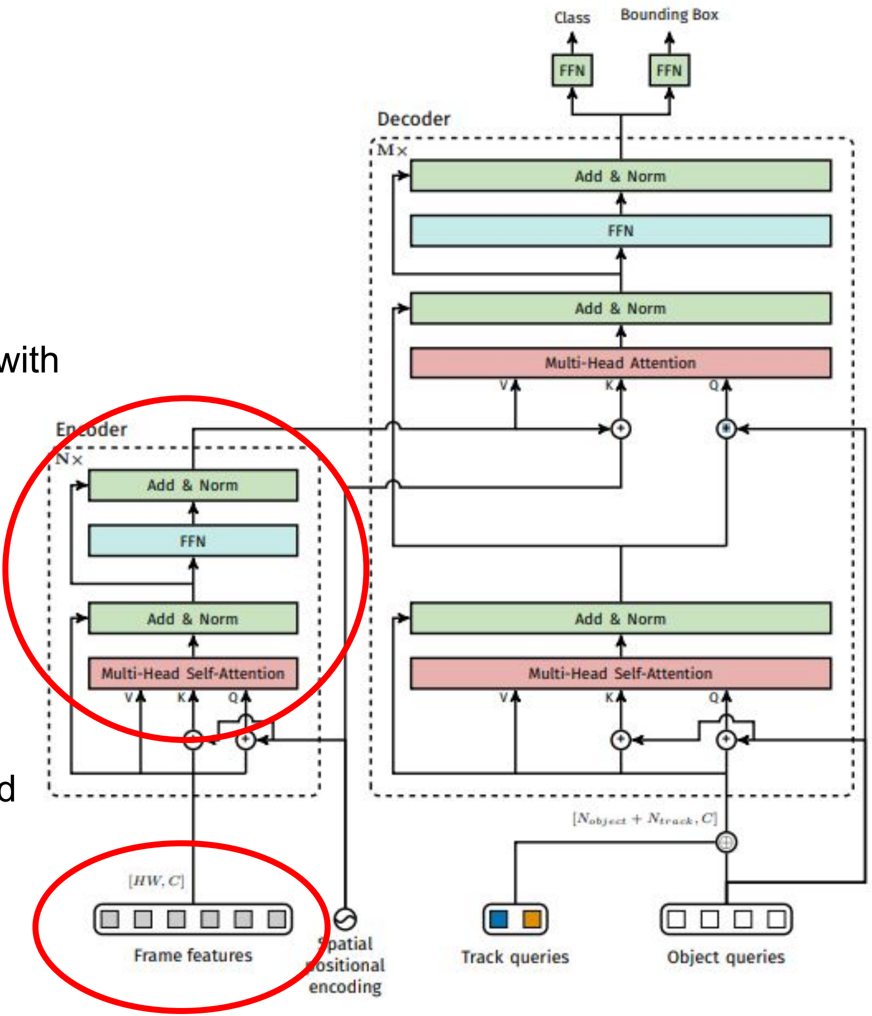


Image features Provided by CNN backbone (ResNet50)

Architecture Overview

Encode Frame features with self-attention

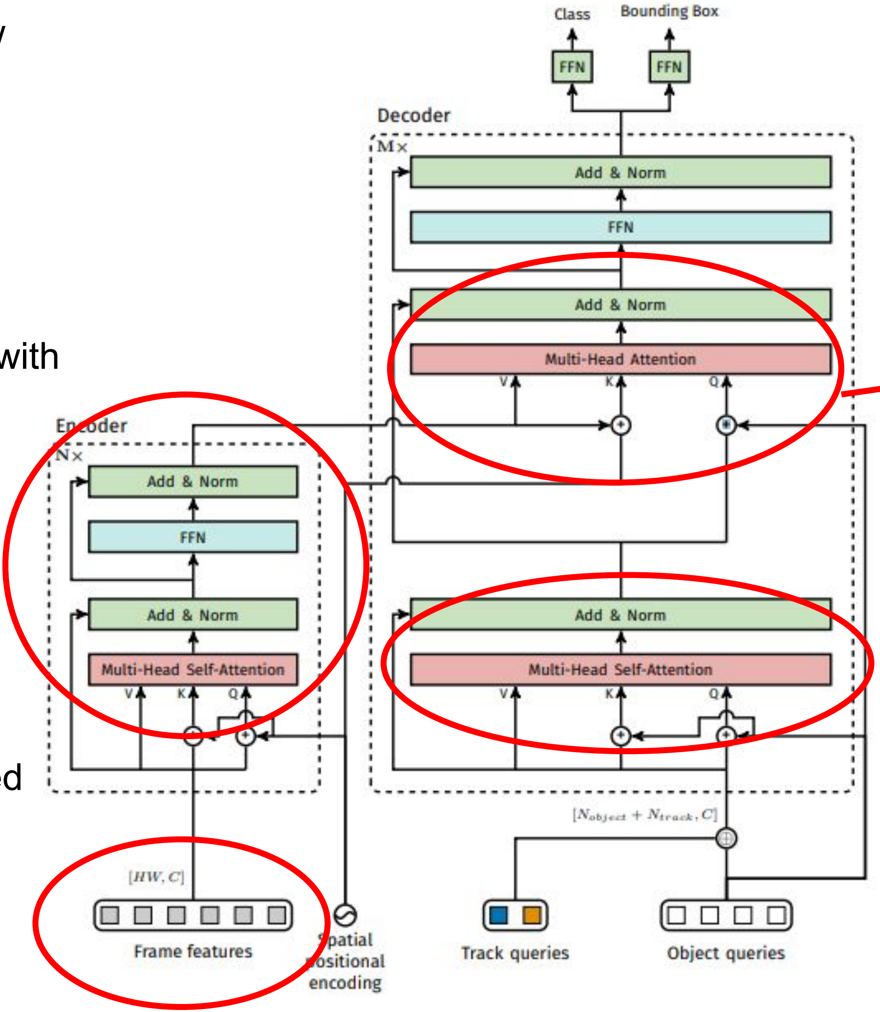
Image features Provided by CNN backbone (ResNet50)



Architecture Overview

Encode Frame features with self-attention

Image features Provided by CNN backbone (ResNet50)

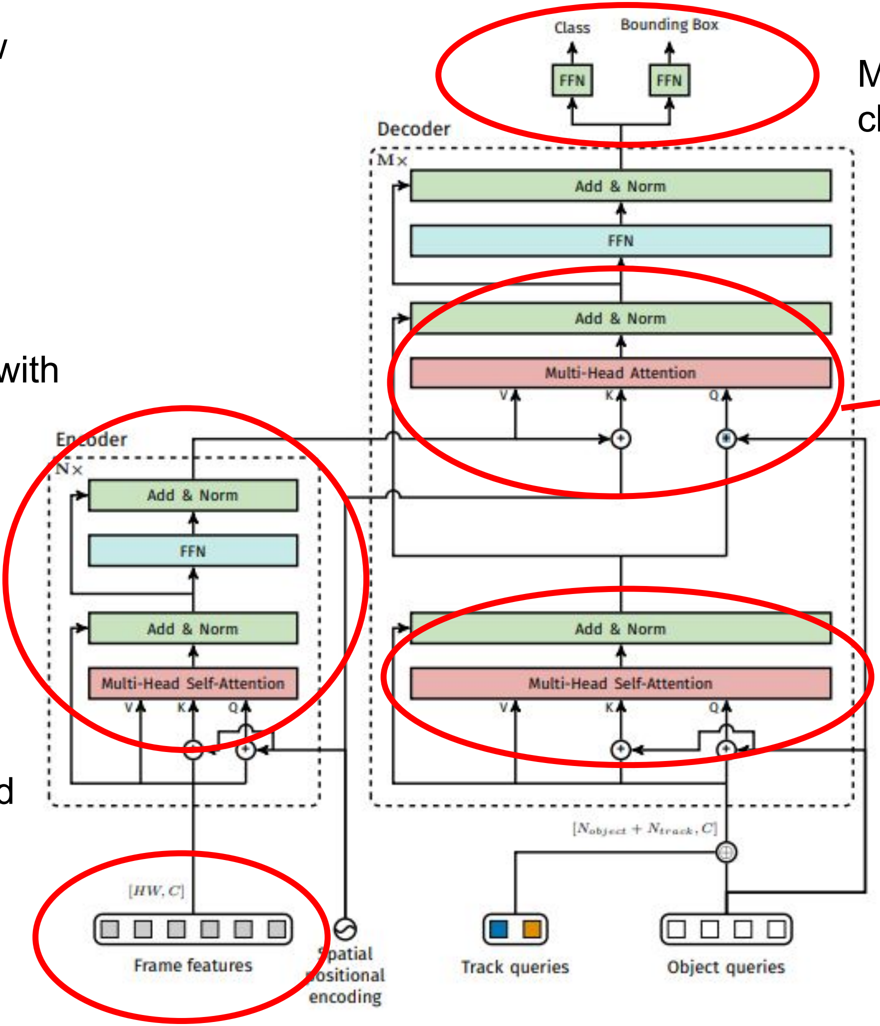


Decode *queries* with self and encoder-decoder attention

Architecture Overview

Encode Frame features with self-attention

Image features Provided by CNN backbone (ResNet50)



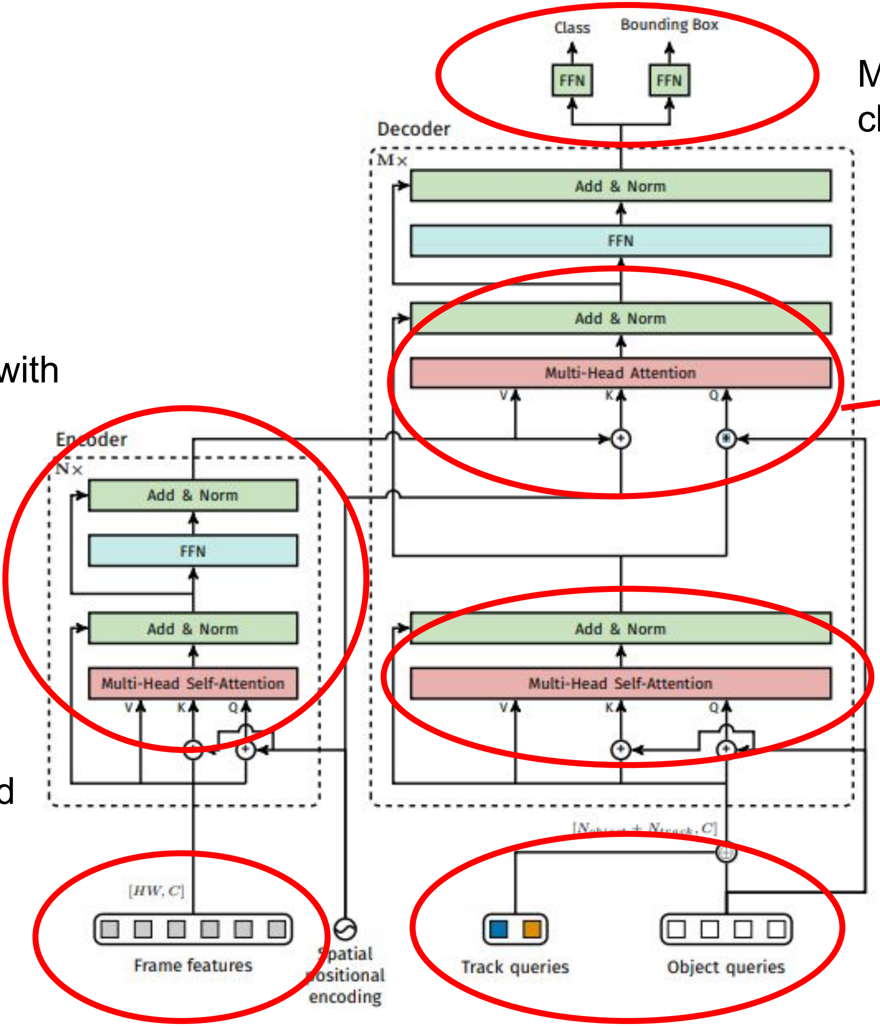
Map Queries to box and class predictions

Decode *queries* with self and encoder-decoder attention

Architecture Overview

Encode Frame features with self-attention

Image features Provided by CNN backbone (ResNet50)



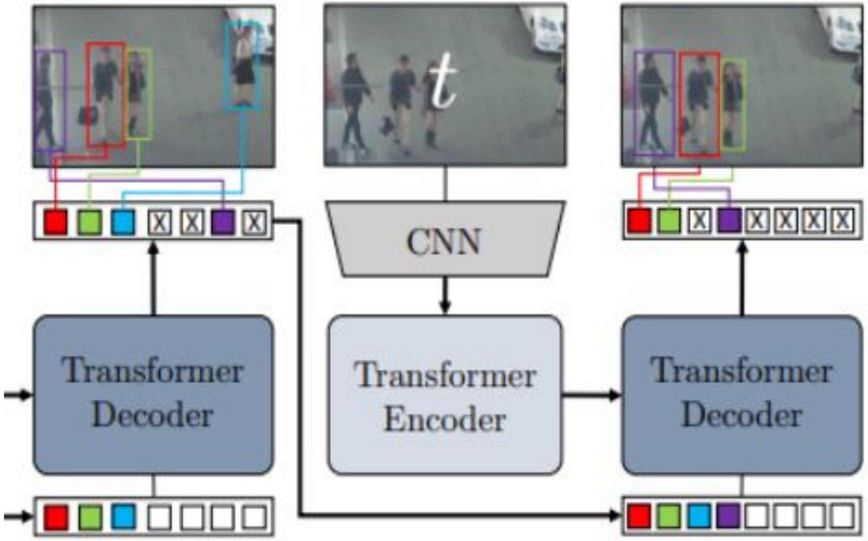
Map Queries to box and class predictions

Decode *queries* with self and encoder-decoder attention

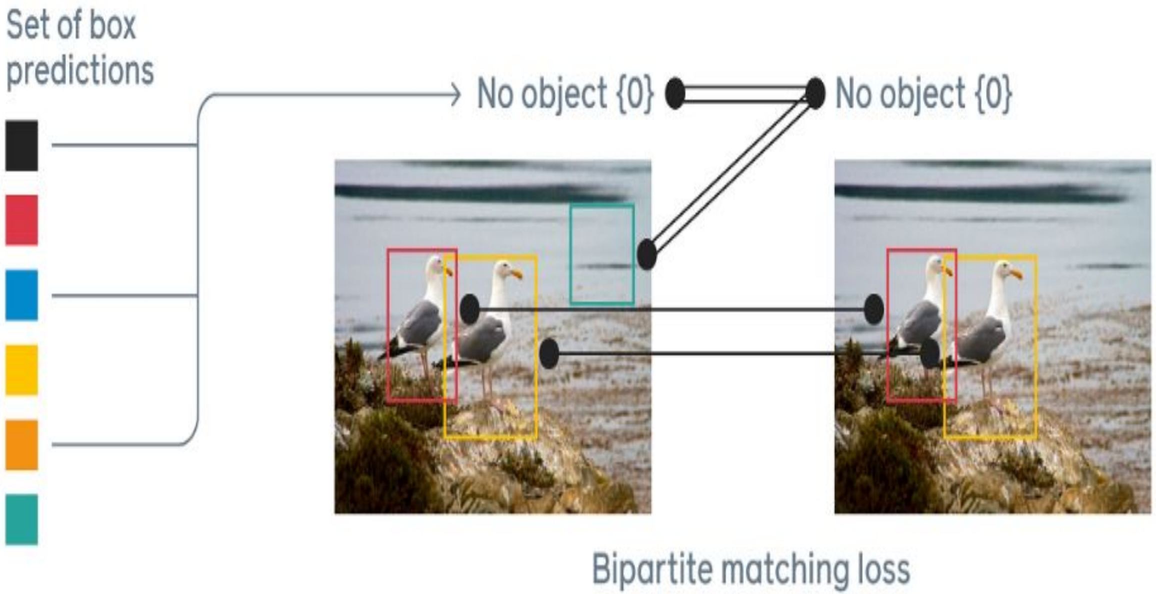
Two types of queries

Track ReID

- Inactive tracks are preserved for a set number of frames "patience window"
- Inactive track queries are reactivated if self attention
- No additional training needed
- Bad for long term occlusions



Training: Bipartite Matching



$K_t \cap K_{t-1}$: Match by track identity k.

$K_{t-1} \setminus K_t$: Match with background class.

$K_t \setminus K_{t-1}$: Match by minimum cost mapping.

$$\hat{\sigma} = \arg\min_{\sigma} \sum_{k_i \in K_{\text{object}}} \mathcal{C}_{match}(y_i, \hat{y}_{\sigma(i)}),$$

$$\mathcal{C}_{\text{match}} = -\lambda_{\text{cls}} \hat{p}_{\sigma(i)}(c_i) + \mathcal{C}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}).$$

$$\mathcal{C}_{\text{box}} = \lambda_{\ell_1} ||b_i - \hat{b}_{\sigma(i)}||_1 + \lambda_{\text{iou}} \mathcal{C}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}),$$

Nicolas Carion, F. Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-toend object detection with transformers. Eur. Conf. Comput. Vis., 2020.

Training: Set Prediction Cost

$$\mathcal{L}_{\text{MOT}}(y, \hat{y}, \pi) = \sum_{i=1}^{N} \mathcal{L}_{\text{query}}(y, \hat{y}_i, \pi).$$

$$\mathcal{L}_{\text{query}} = \begin{cases} -\lambda_{\text{cls}} \log \hat{p}_i(c_{\pi=i}) + \mathcal{L}_{\text{box}}(b_{\pi=i}, \hat{b}_i), & \text{if } i \in \pi \\ -\lambda_{\text{cls}} \log \hat{p}_i(0), & \text{if } i \notin \pi. \end{cases}$$

Example Performance

Summary

- Object Tracking
  - Object Recognition and understanding of temporal relationships between objects
- Recurrent Neural Networks
  - Neural network designed to relate information between sequential inputs
- Transformers
  - New methods of analyzing and understanding relationships between sequential inputs and outputs.
- Trackformer: Multi-Object Tracking
  - Uses attention to both detect AND track objects through “queries”

DeepRob

[Student] Lecture 16

by Mohammed Guiga, Danny Langan, Pranav Julakanti

Object Tracking, Transformer Architecture

University of Michigan and University of Minnesota