# Transformers

## Attention Is All You Need (2017)

Getting away from RNNs ad LSTM in NLP.

*LSTM*: use the hidden state from the previous input and plugs it in the next model and so on.

*Attention*: increases performance of RNNs, by paying attention to the decoder back at the hidden states from previous encodings. The decoder outputs keys that index the hidden states. You don’t need the whole recurrence, but can look directly to certain tokens.

***Architecture:***

* Encoder:
  + Discovers interesting things.
  + Input embedding = builds **K**ey/**V**alue pairs.
  + Keys are a way to index the values.
* Decoder:
  + Output embedding = builds **Q**ueries (like attributes())
* Together multi-head attention.

Uses attention gates to look back at hidden representations from previous inputs.

***Attention mechanism*:**

* Encode the input into a vector and choose which part we pay more attention. Where each attention has:
* Key (vectors attached to input tokens)
* Value (vectors )
* Query (vectors attached to each token’s top layers).

### Method:

* One step prediction instead of recurrent prediction for every token (RNN).
* We loose the sequence so we need a positional encoding.
* Each token aggregates information from all other token with special attention to some tokens, layers after layers.
* We compute the inner product between each token’s key and query vectors to a softmax exponential functions which gives a histogram with max prob.

### Pros:

* Reduces path length (better for computation steps).
* Better performance than RNNs.

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018)

Uses Attention from transformers from left-to-right and right-to-left.

### Pros:

* Looking at both directions simultaneously.
* Use self-supervision masking for pre-training.

## DETR: End-to-End Object Detection with Transformers (2020)

Uses CNN and Transformer to do object detection.

### Architecture:

* Backbone:
  + CNN: encoder gives set of features.
  + Positional encoding.
* Encoder (feature mapping):
  + Input: features and positional encoding.
  + Transformer encoder is a sequence processing unit. And since an image in not a sequence. The input sequence is the unrolled channels and height x width from the image.
  + Transformer: unroll the image into height and width. The transformer gathers information in the sequence from any other point in the sequence.
  + Attention: each channel input can attend to each other position in the sequence.
  + Quadratic: Every instance can look at another (height\*width)²
  + Output: Same size and shape out of the transformer.
* Decoder (Attention mechanism):
  + Input: intake object queries (N vectors) that are learned.
  + Algorithm: train N different embeddings to describe the data (learned) and ask questions by looking at particular feature maps (attention maps).
  + Output: same size and shape as input. (N vectors).
  + Feed Forward Network (FFN).
* Transformer: encoder-decoder.
  + Outputs: set of box predictions (class, bbox)
* Bipartite matching loss:
  + Prevent from classifying multiple times the same object.

### Pros:

* Simpler and better results than F-RCNN.

# Computer Vision Transformers

## Vision Transformers (ViT): An Image Is Worth 16x16 Words (2020)

Uses attention mechanism that are quadratic (n²) since every token connects to every pair (good for NLP but too large in memory for images). Images have many pixels, such as Imagenet: (125²)² is impossible to fit into memory with Transformers’ quadratic Attentions.

Solution could use local attentions mechanisms (similar to CNNs). Lets use Global Attentions by going over images patches (16x16)²

### Pros:

* Transformers can perform very well for image classification tasks when applied to sequence of image patches.
* Require fewer computational resources to train than CNNs.
* More general bias than CNNs and LSTM (more global approach at solving a problem).

### Methods:

#### Transformer:

* Uses attention which is a quadratic operation. It calculates a pair wise inner product between each pair of tokens in the sequence.
* Images have many pixels. Thus, many pair wise operations.
* They use global attention by dividing images into patches of 16x16 and positional embedding. Then, unroll the patches into a vectors and multiply with the Embedding vector (add the positional embedding to it).
* Feed this to the Transformer.
* Special input: learnable embedding (\*= CLS) not associated with any patch.
* Last layer: we compute the MLP Head.

# DINO: Emerging Properties in Self-Supervised Vision Transformers (2021)

Self-**Di**stillation with **no** labels: Self-attention from a Vision Transformer with 8 x 8 patches trained with no supervision. Student – teacher approach (self-distillation).

### Pros:

* Features are valuable.
* Self-supervised learning and ViT goes well together.

### Methods:

#### Vision Transformers

* Divide an image into patches and unroll those patches in a sequence.
* Use a FC layer to get the embedding.
* Use a Transformer (use a special token “cls”).
* Transformer will give an output for every input token.
* CLS will aggregate all the information from all the visual tokens in the image.

#### Self-supervised learning

Train a model without labels. It learns the labels by itself.

* *Contrastive learning (not used in this paper)*: is a machine learning technique used to learn the general features of a dataset without labels by teaching the model which data points are similar or different. By looking at similar (+ pair) or different (-pair) of classes of objects it can learn features. For similar images it uses data augmentation to change one of the pair.

The goal is to output similar representation embeddings for similar images. Loss function uses cosine angle metric distance between vectors. The training algorithm knows if pairs are similar but doesn’t know which class are the images.

* **Architecture and Data Augmentation (with BYOL)**

Make 2 different versions of the same image (x1 and x2) from x with data augmentation. Goal is to have the same representation embedding from 2 different augmentations on the same crops from the same image with only have positive pairs (no negative samples):

* *Global Crops: >50% of the image.*
* *Local Crops: < 50% of the image.*

*Distillation***:** uses 2 different models:

* *Student:* we train the student.
  + *Local crops.*
* *Teacher:* teacher model is built from the student.
  + *Global crops.*
* *EMA:* Exponentially Moving Average between student and teacher model.
* Both student and teacher predict the same representation. Although the teacher sees more from the image.

By looking at the Attention maps from the CLS token, it gives semantic segmentations of objects.

### Intuition: Why so good results?

#### Augmentations

Prior bias told to the model that doesn’t matter. The model should not pay attention to features that have been augmented.

#### Dataset

How the dataset is constructed. Is there a clear object in the image? The way the data is captured gives an intuition to the model at what to look for.

# VTN: Video Transformer Network (Theator, 2021)

### Pros:

* Transformer-based framework for video recognition using vision transformers.
  + Attention on the whole video sequence information.
* Trains 16 x faster and predicts 5 x faster compared to other SOTA models.
* Suitable for long (surgical) videos.