Self-Attention for Vision

CS4245 "Seminar Computer Vision by Deep Learning"

R. Bruintjes, June 2nd 2021

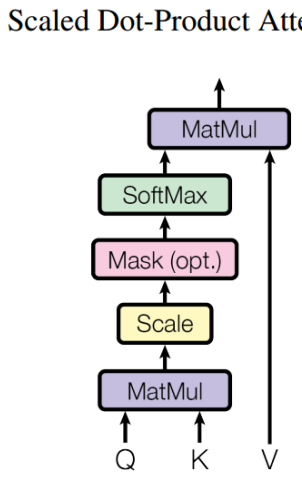
This lecture

- Recap of self-attention/Transformers from DL course
- Problems and solutions in applying Transformers in vision
- Application: object detection

Recap: Concepts

Self-attention

- Operation to use in a Deep Net
- Compare to: convolution, feed forward



Transformer

- Architecture
- Sequence-to-sequence tasks
- Encoder & decoder
- Uses self-attention
- Compare to: ResNet, RNN

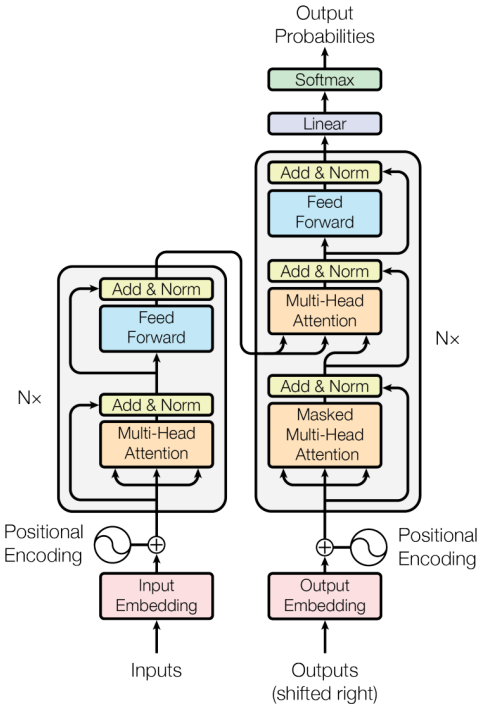


Figure 1: The Transformer - model architecture.

Figures from Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, 30. https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Self-attention

From Deep Learning course

Example self-attention powered architecture

- Data: a sentence = sequence of words
- Task: predict next token = next word
- Representation: token is a R^{d} vector

How to predict next word

- Self-attention layer: *some weighted combination of all tokens*
- MLP: just on the single token

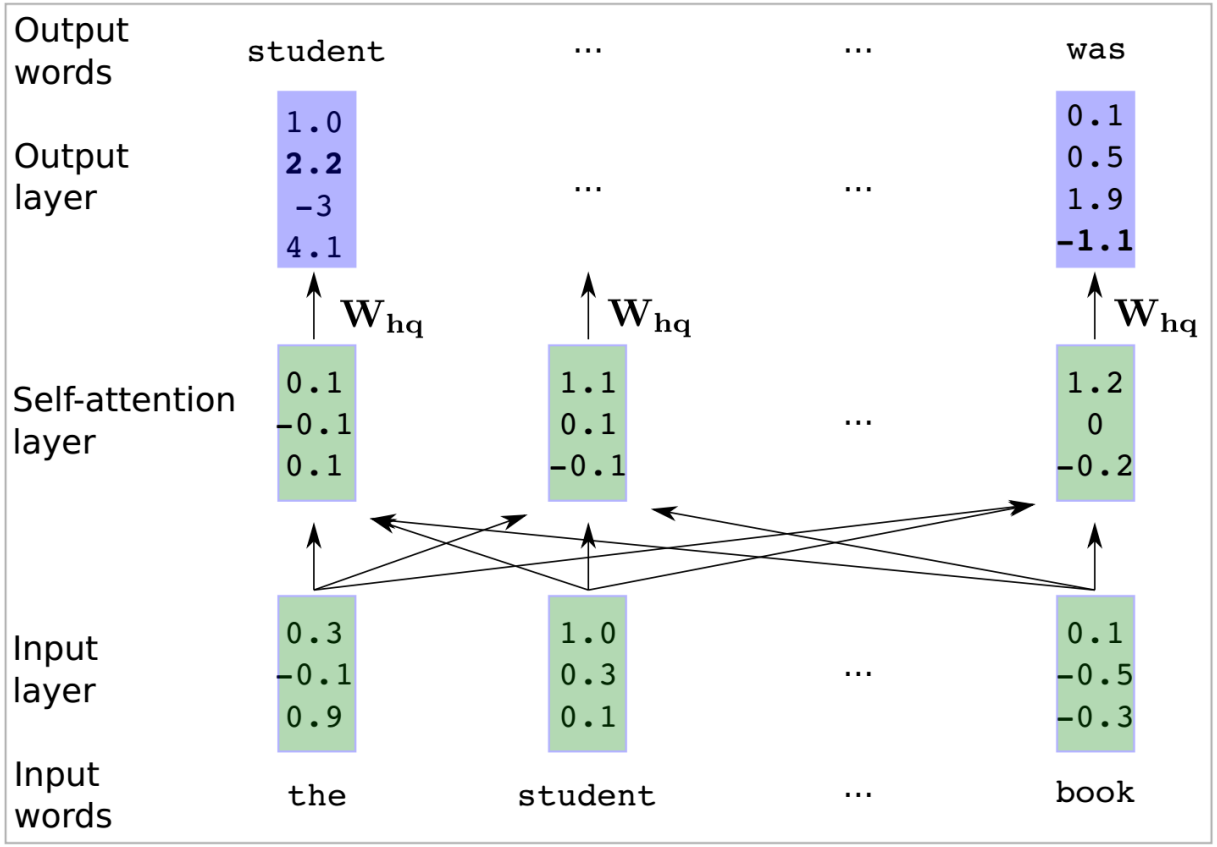


Figure from Lecture 7 of CS4240 Deep Learning, by Jan van Gemert.

Query - Key - Value

Movie database retrieval

- Query: I'm looking for *an action movie from 2008*
- Database
  - Value: **Interstellar**; Key: *action movie, 2014*
  - Value: **Shrek**; Key: *comedy, 2001*
  - Value: **The Dark Knight**; Key: *action movie, 2008*

Query - Key - Value

Movie database retrieval

- Query: I'm looking for *an action movie from 2008*
- Database
  - Value: **Interstellar**; Key: *action movie, 2014*
  - Value: **Shrek**; Key: *comedy, 2001*
  - Value: **The Dark Knight**; Key: *action movie, 2008*
- Query matches key, retrieves value: "The Dark Knight"

Query - Key - Value

Movie database retrieval

- Query: I'm looking for *an action movie from 2008*
- Database
  - Value: **Interstellar**; Key: *action movie, 2014*
  - Value: **Shrek**; Key: *comedy, 2001*
  - Value: **The Dark Knight**; Key: *action movie, 2008*
- Query matches key, retrieves value: "The Dark Knight"

➔ Self-attention = "soft, weighted retrieval"

"Soft, weighted retrieval"

- We have a "query" token
  - "What are we looking for?"
  - Some d-dimensional feature

$$q \in \mathbb{R}^d$$

"Soft, weighted retrieval"

- We have a "query" token $q \in R^d$
  - "What are we looking for?"
  - Some d-dimensional feature
- For the database of size N we have key tokens and value tokens $k_j, v_j \quad j \in [0, N]$

"Soft, weighted retrieval"

- We have a "query" token $q \in R^d$
  - "What are we looking for?"
  - Some d-dimensional feature
- For the database of size N we have key tokens and value tokens $k_j, v_j \quad j \in [0, N]$
- Compare the query q against "key" tokens k_j.
  - We use softmax similarity: $\text{sim}(q, k_j) = \text{softmax}(q^T k_j) = \frac{\exp(q^T k_j)}{\sum_j \exp(q^T k_j)}$
  - Each similarity represents **how much the query matches the key**

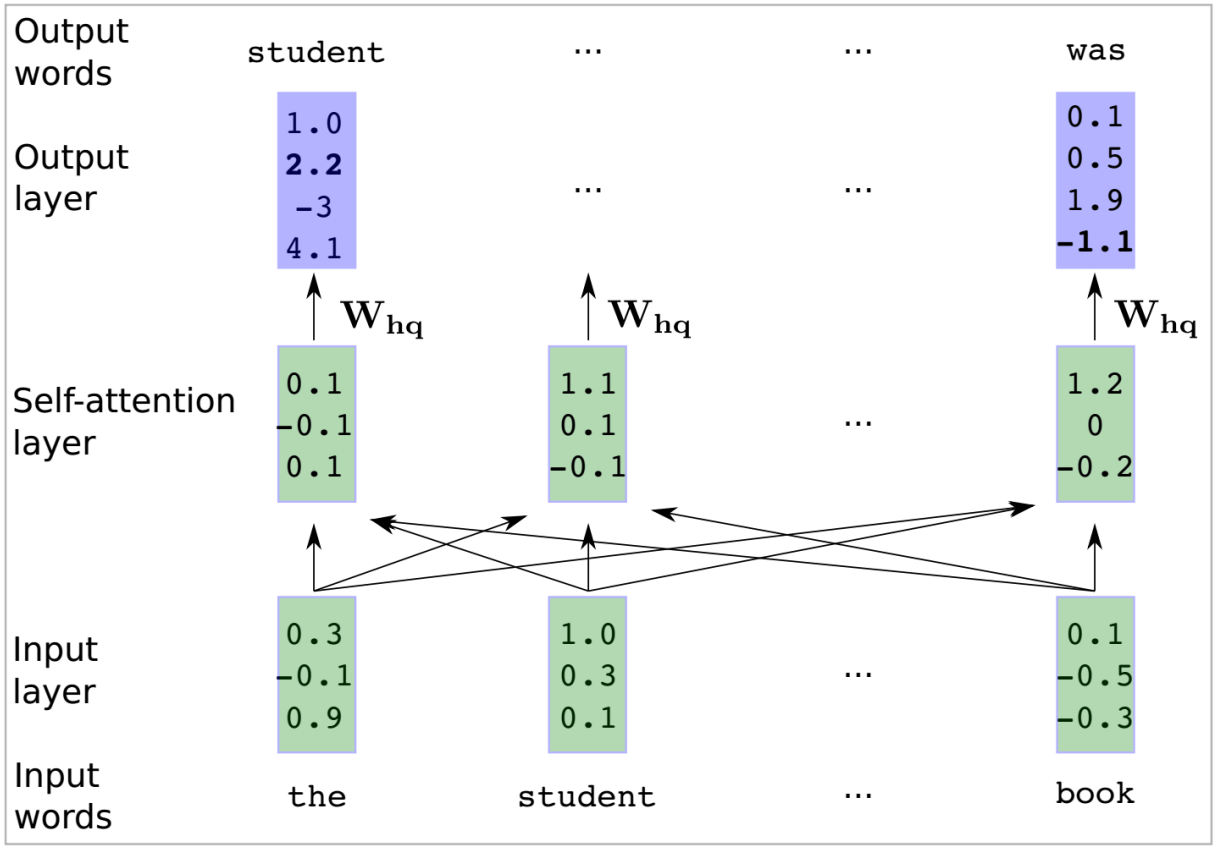
"Soft, weighted retrieval"

- We have a "query" token $q \in R^d$
  - "What are we looking for?"
  - Some d-dimensional feature
- For the database of size N we have key tokens and value tokens $k_j, v_j \quad j \in [0, N]$
- Compare the query q against "key" tokens k_j.
  - We use softmax similarity: $\text{sim}(q, k_j) = \text{softmax}(q^T k_j) = \frac{\exp(q^T k_j)}{\sum_j \exp(q^T k_j)}$
  - Each similarity represents

    how much the query matches the key
- Output is a sum over *values* weighted by similarity with *key*: $y = \sum_j (\text{sim}(q, k_j) \cdot v_j)$
  - ➔ a "soft" database retrieval

Self-attention

Use linear projection to create queries, keys and values:

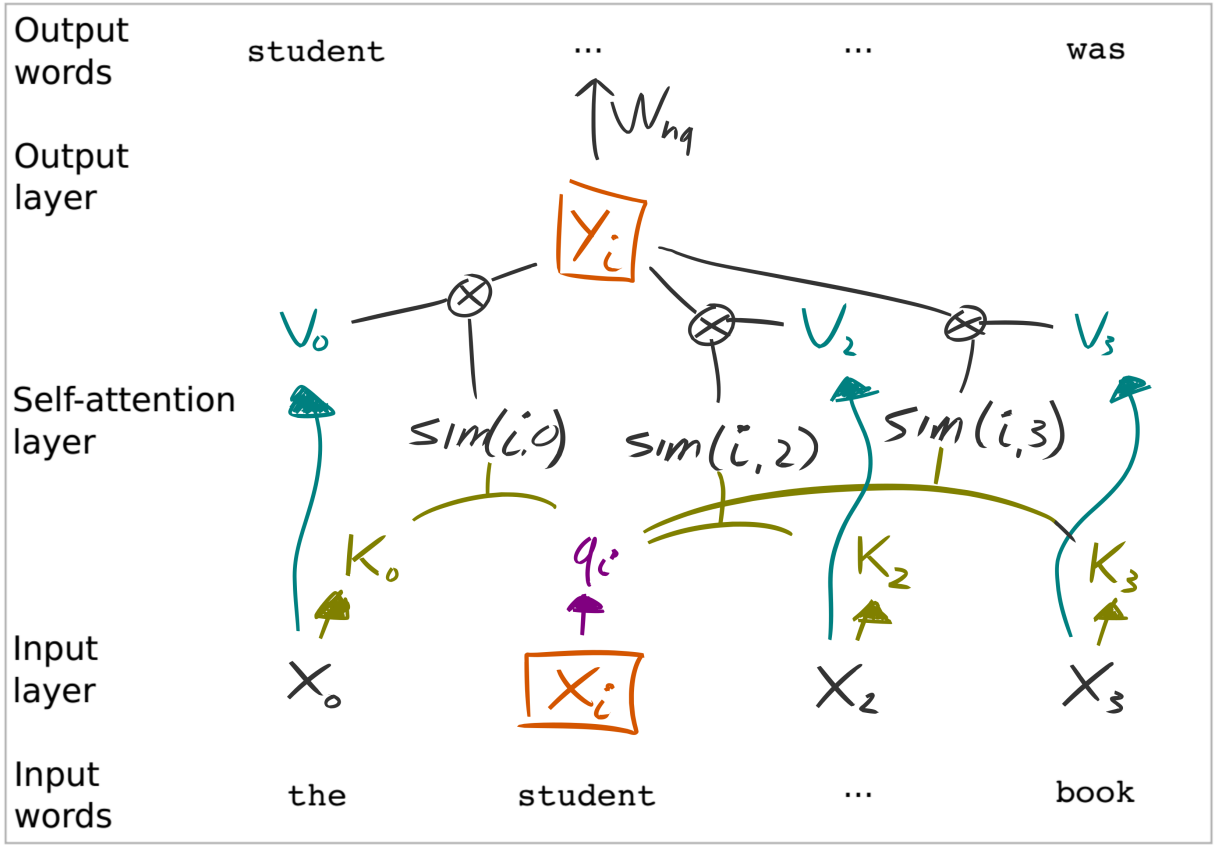


Self-attention

Use linear projection to create queries, keys and values:

$$q_i = W_q x_i + b_q$$
$$k_i = W_k x_i + b_k$$
$$v_i = W_v x_i + b_v$$
$$y_i = \sum_j \mathrm{sim}(q_i, k_j) \cdot v_j$$

(Note: i is included in j, but not shown)



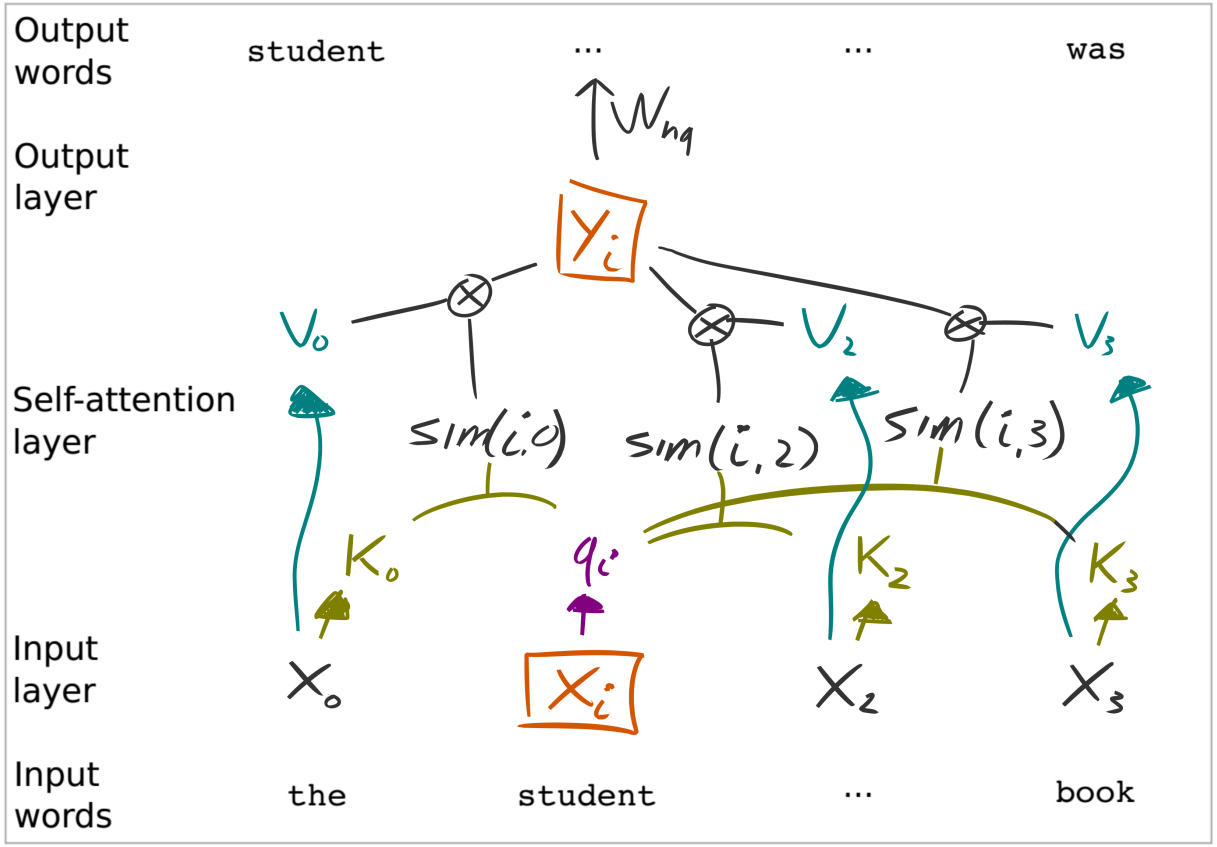
Self-attention

Use linear projection to create queries, keys and values:

$$q_i = W_q x_i + b_q$$
$$k_i = W_k x_i + b_k$$
$$v_i = W_v x_i + b_v$$
$$y_i = \sum_j \mathrm{sim}(q_i, k_j) \cdot v_j$$

(Note: i is included in j, but not shown)

Q: Complexity?



Self-attention

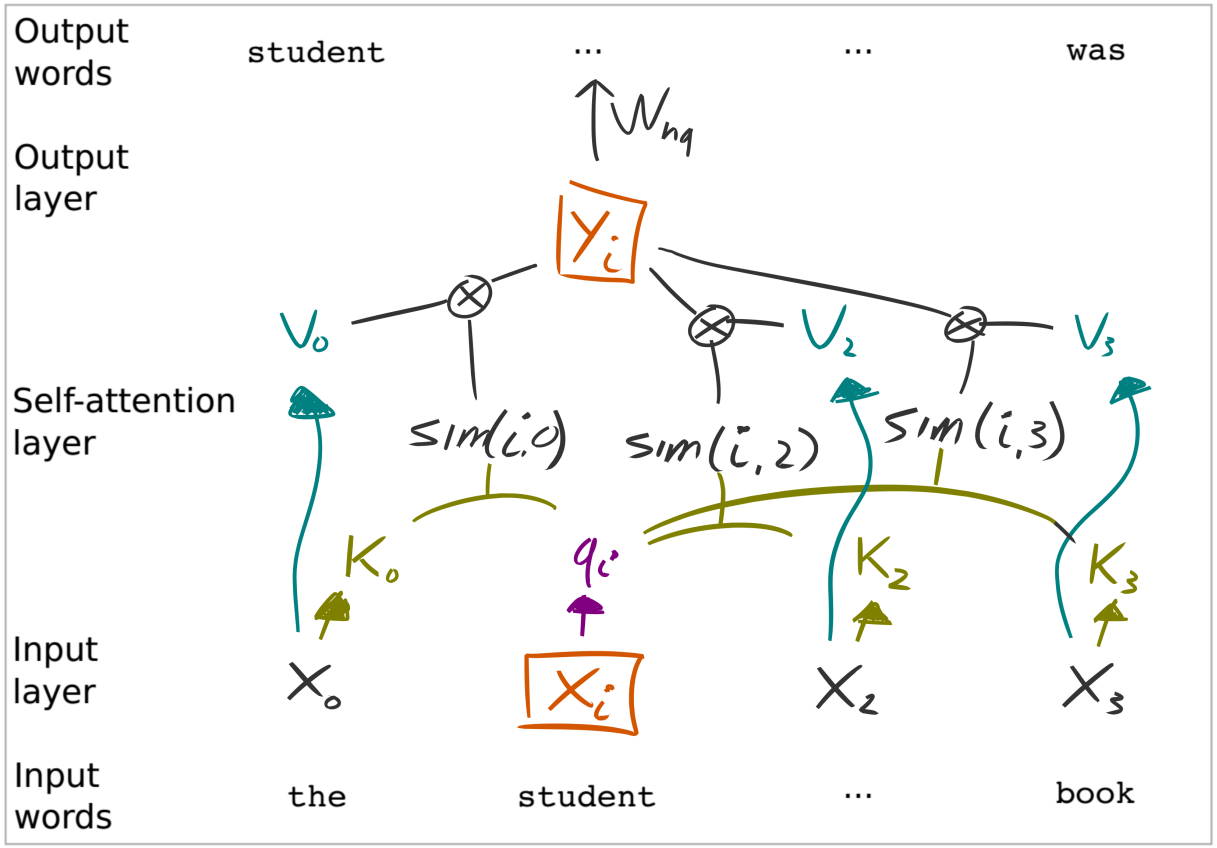
Use linear projection to create queries, keys and values:

$$q_i = W_q x_i + b_q$$
$$k_i = W_k x_i + b_k$$
$$v_i = W_v x_i + b_v$$
$$y_i = \sum_j \mathrm{sim}(q_i, k_j) \cdot v_j$$

(Note: i is included in j, but not shown)

Q: Complexity?

A: $O(N^2 d)$, because of $\mathrm{sim}(q_i, k_j)$



Transformers

Transformer Vaswani et al. 2017 =

- "Transformer layer"
  - Self-attention
  - Feed Forward (AKA MLP)
  - Skip-connections & normalization
- Positional encodings
- Used in architecture with encoder & decoder

These components are used interchangeably and selectively!

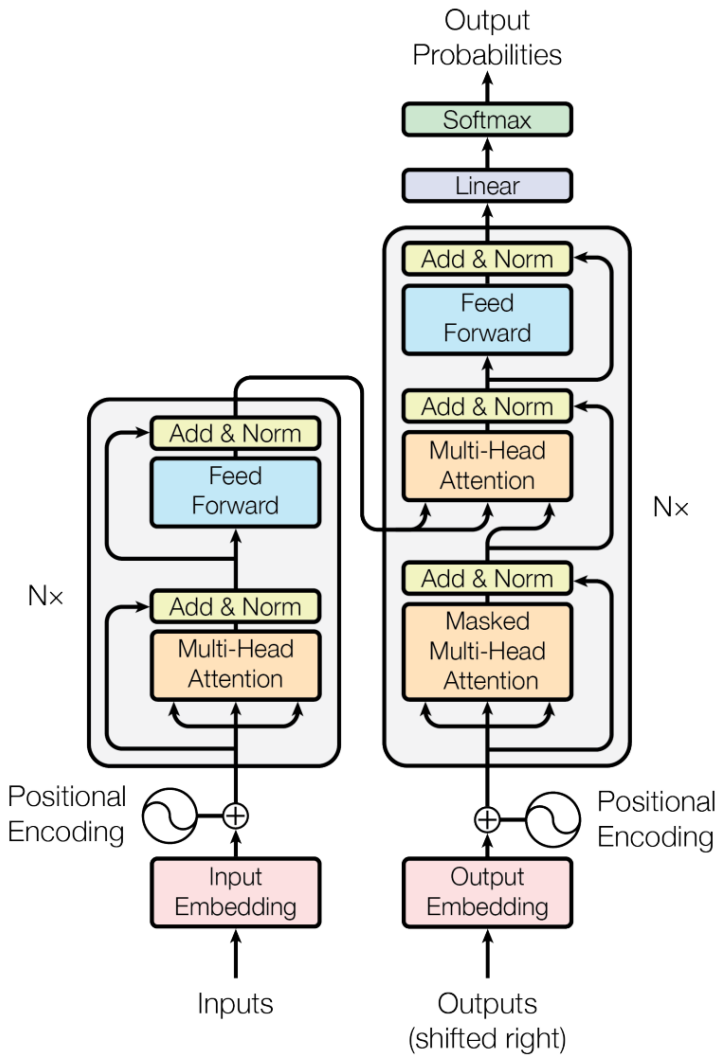


Figure 1: The Transformer - model architecture.

Figure from Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, 30. https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Tasks to use a Transformer with

In Natural Language Processing (NLP): sentences = sequences of words

- Words are discrete, semantic things
- Words can be represented with a vector embedding $x \in \mathbb{R}^d$, e.g. Word2Vec

Tasks to use a Transformer with

In Natural Language Processing (NLP): sentences = sequences of words

- Words are discrete, semantic things
- Words can be represented with a vector embedding $x \in \mathbb{R}^d$, e.g. Word2Vec

In Computer Vision: data is images: collection of pixels

Q: How could we use a Transformer for images?

Tasks to use a Transformer with

In Natural Language Processing (NLP): sentences = sequences of words

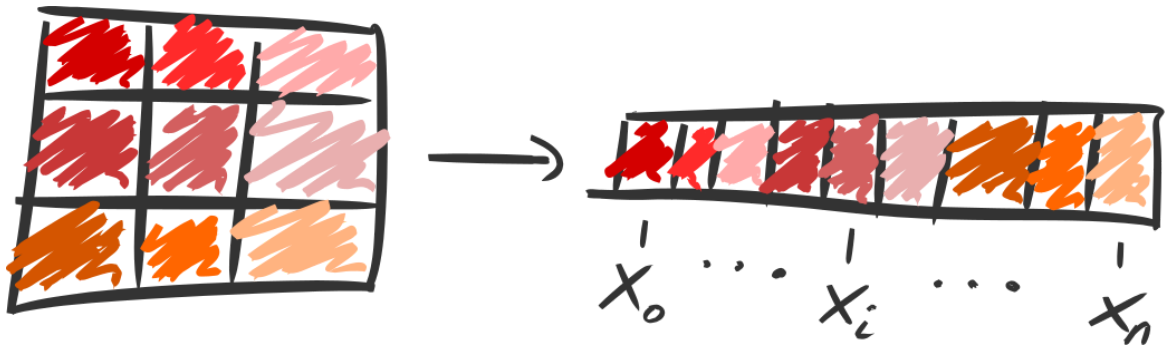
- Words are discrete, semantic things
- Words can be represented with a vector embedding $x \in \mathbb{R}^d$, e.g. Word2Vec

In Computer Vision: data is images: collection of pixels

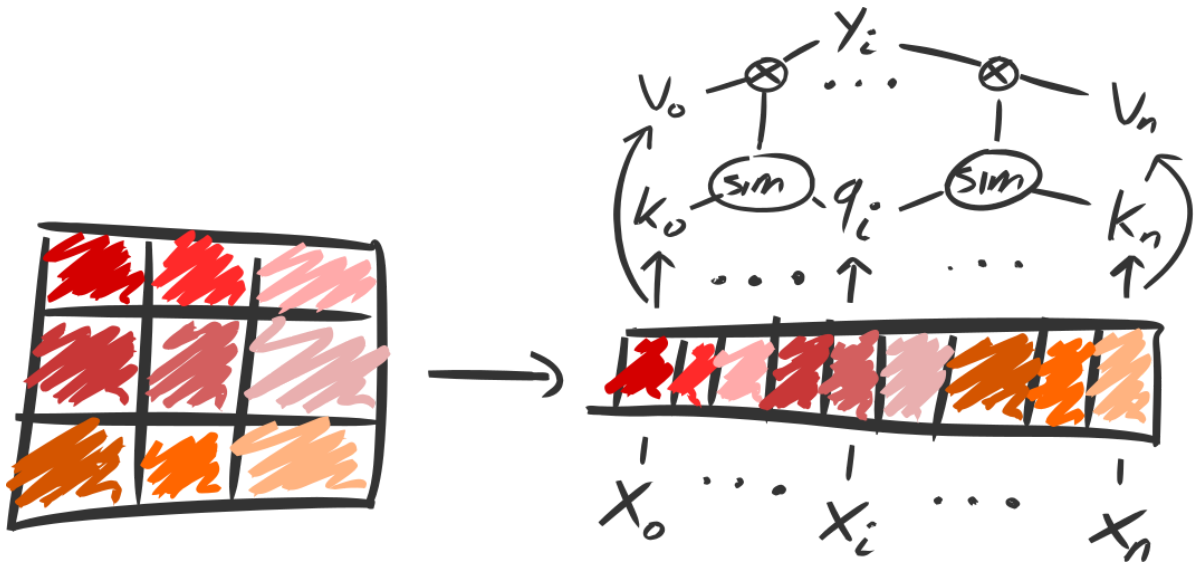
Q: How could we use a Transformer for images?

A: Treat image as a (2D) sequence!

Transformers for Vision

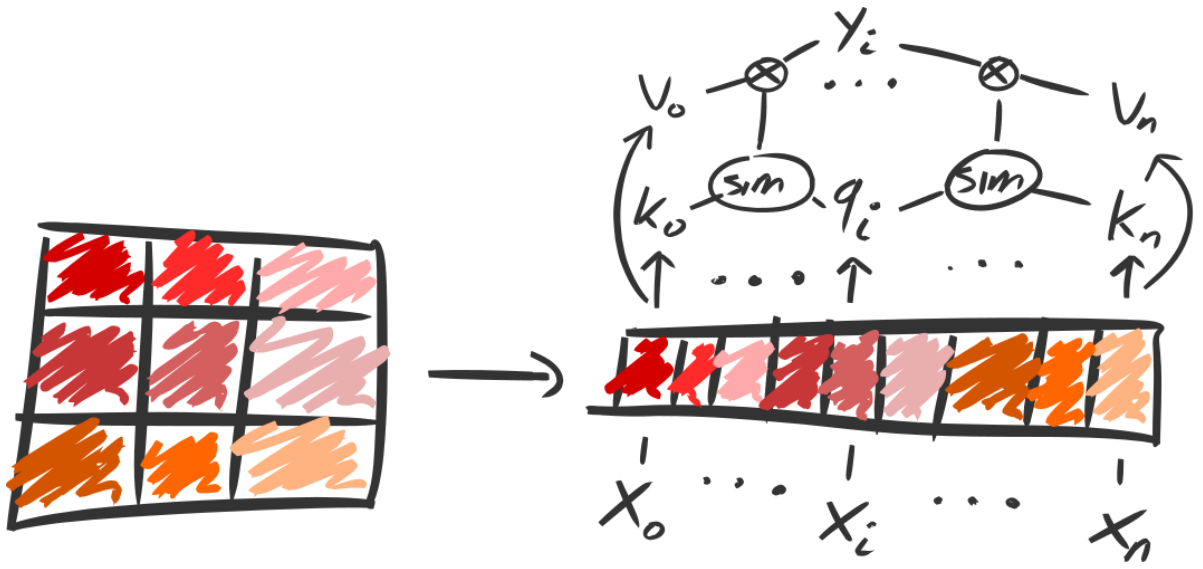


Transformers for Vision



Transformers for Vision

Q: Can anyone see a problem with using self-attention for pixels?

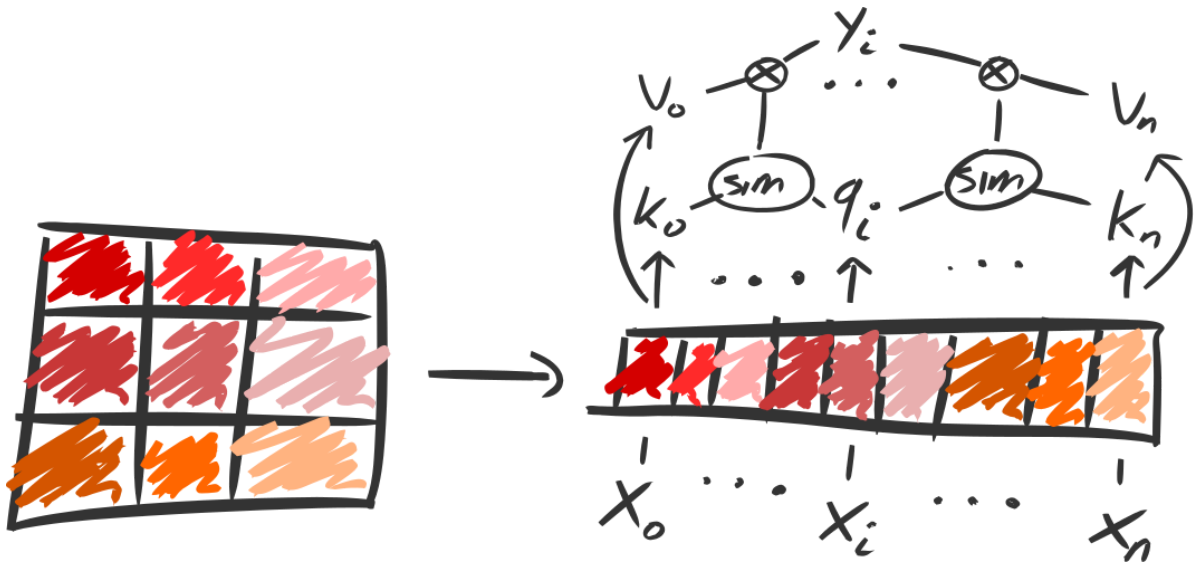


Transformers for Vision

Q: Can anyone see a problem with using self-attention for pixels?

A: Complexity $O(N^2 d)$, but with N = amount of pixels!

E.g. ImageNet resolution, $224 \times 224 = 50176$ pixels
➔ $50176^2 = 2.5 \times 10^9$



Transformers for Vision

Q: How do we solve this?

Transformers for Vision

Q: How do we solve this?

- ViT
- SASA

Transformers for Vision

Q: How do we solve this?

- ViT
- SASA

Let's go ahead

Solution 1: compress pixels

Vision Transformer (ViT)

Dosovitskiy et al. (2020)

- Use a convolution to compress patches to tokens
  - 16×16 convolution with stride 16.
- Then use regular Transformer encoder
- (-) Needs pre-training on JFT (303M images)

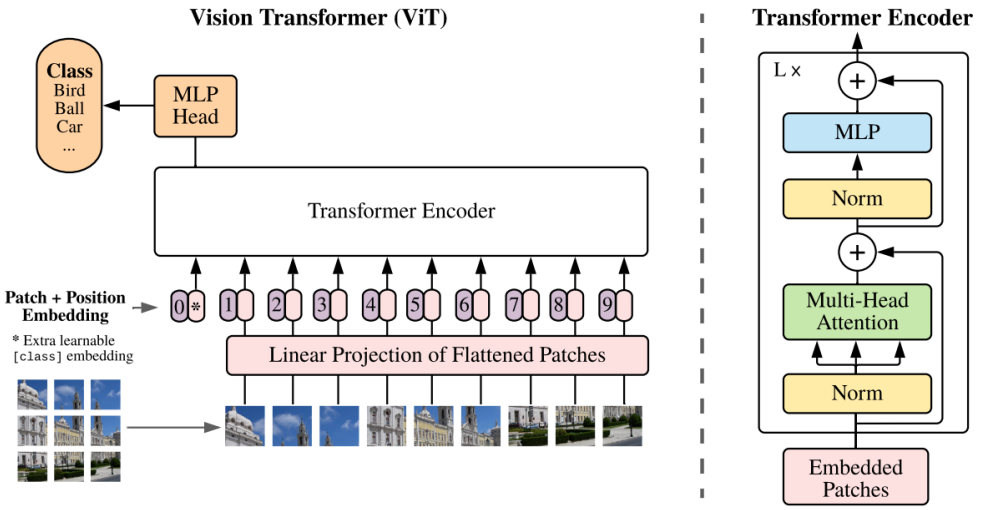


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Take me back

Solution 2: don't use all pixels

Stand-Alone Self-Attention (SASA)

Ramachandran et al. (2019)

- Query = center pixel
- Keys & values = *local window around query*
- Local operator, much like convolution!
  - $y_i = \sum_{j=-k}^{k} \mathrm{sim}(q_i, k_{i+j}) \cdot v_{i+j}$
- ...but still needs downsampling in architecture.

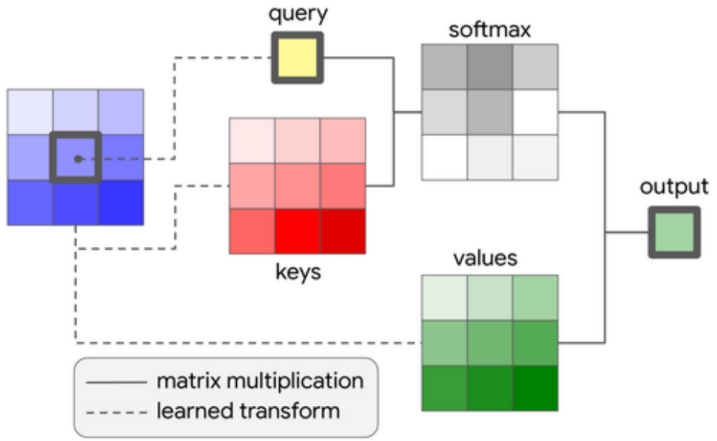


Figure 3: An example of a local attention layer over spatial extent of $k = 3$.

Ramachandran, P., Parmar, N., Vaswani, A., Bello, I., Levskaya, A., & Shlens, J. (2019). Stand-Alone Self-Attention in Vision Models. Neural Information Processing Systems.

Take me back

Solution 3: approximate softmax attention

Performers Choromanski et al. (2020)

- Softmax is a kernel between q and k
- Use a sort of "inverse kernel trick":
  - Map k to random features, to match kernel formulation
  - Do $\mathrm{k} \times \mathrm{v}$ before $\cdots \times \mathrm{q}$
- Great paper, but too difficult for mainstream apparently.

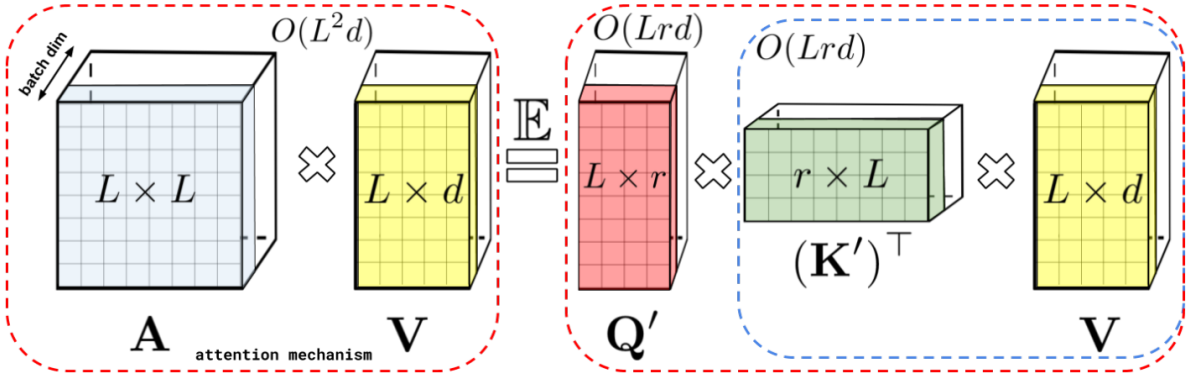


Figure 1: Approximation of the regular attention mechanism $\mathbf{AV}$ (before $\mathbf{D}^{-1}$-renormalization) via (random) feature maps. Dashed-blocks indicate order of computation with corresponding time complexities attached.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2020). Rethinking Attention with Performers. ArXiv:2009.14794 [Cs, Stat]. http://arxiv.org/abs/2009.14794

Take me back

State of the field

- ViT: fall 2020
- Now: one or two waves of research on top
- Needs time to settle

State of the field

- ViT: fall 2020
- Now: one or two waves of research on top
- Needs time to settle

Open questions

- Can we train without extra "tricks"?
- Benefits of ConvNet-like architectures?
- Do we even need self-attention?
  - Fourier transform Lee-Thorp et al. (2021)
  - Only MLPs Tolstikhin et al. (2021)

Lee-Thorp, J., Ainslie, J., Eckstein, I., & Ontanon, S. (2021). FNet: Mixing Tokens with Fourier Transforms. arXiv preprint arXiv:2105.03824. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., ... & Dosovitskiy, A. (2021). MLP-Mixer: An all-MLP architecture for vision. arXiv preprint arXiv:2105.01601.

Application: object detection

DEtection TRansformer (DETR)

Carion et al. (2020)

- Vanilla Transformer architecture
- CNN preprocessing
- Transformer encoding (on CNN feature map)
- Transformer decoder + small MLP over (learned) *object queries*
- Use Hungarian algorithm to match proposals to groundtruths

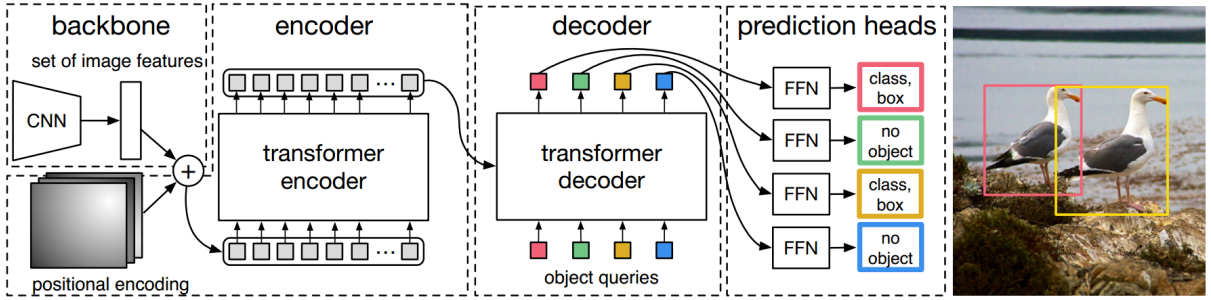


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. ArXiv:2005.12872 [Cs]. http://arxiv.org/abs/2005.12872

Application: object dete

DEtection TRansformer (DETR)

Carion et al. (2020)

- Vanilla Transformer architecture
- CNN preprocessing
- Transformer encoding (on CNN feature map)
- Transformer decoder + small MLP over (learned) *object queries*
- Use Hungarian algorithm to match proposals to groundtruths

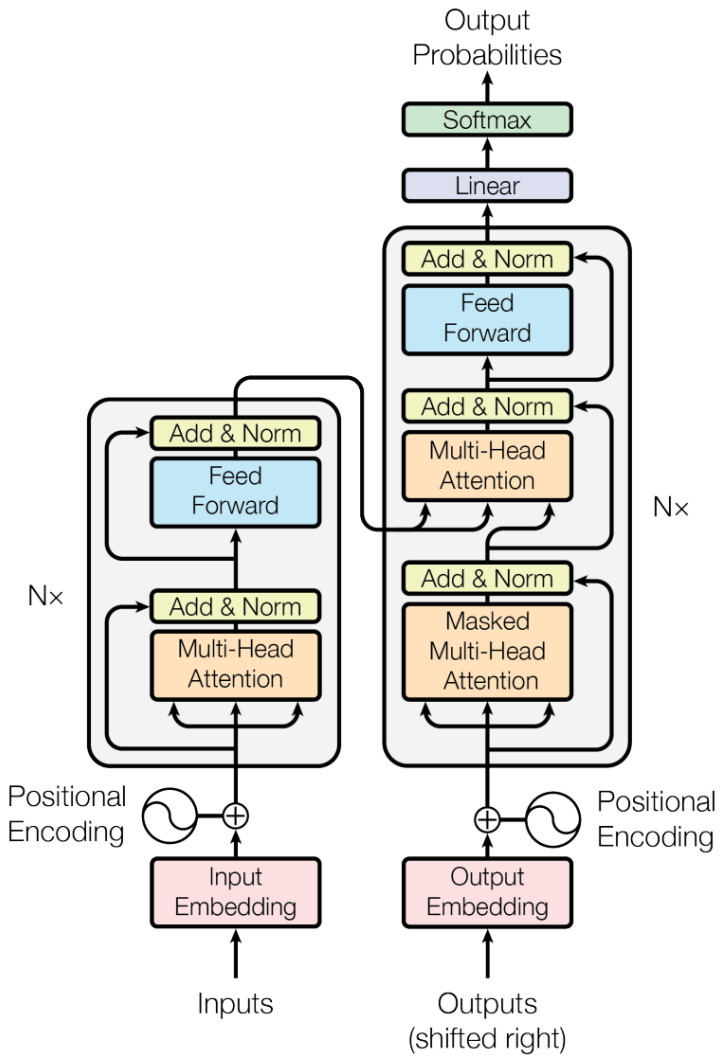


Figure 1: The Transformer - model architecture.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. ArXiv:2005.12872 [Cs]. http://arxiv.org/abs/2005.12872

Application: object detection

Pros & cons

- (+) Straightforward architecture
- (+) No need for NMS, negative sampling, etc.
- (-) Not yet tuned as well as state-of-the-art

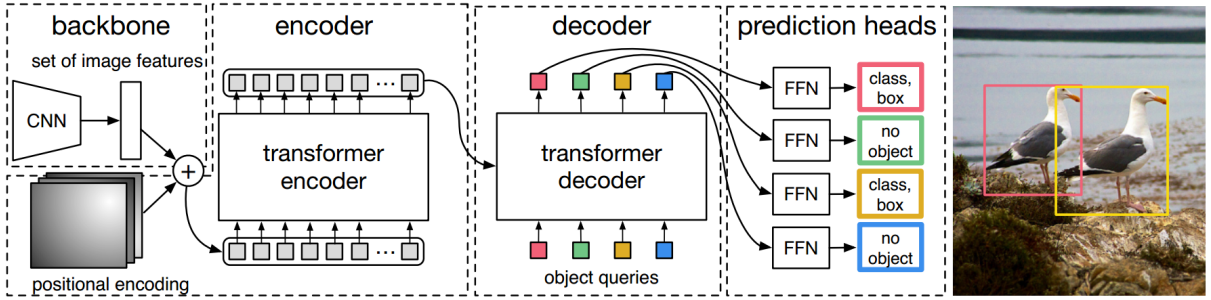


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. ArXiv:2005.12872 [Cs]. http://arxiv.org/abs/2005.12872

Conclusion

Recap

- Self-attention: global operation for sequences
- Transformer: encoder-decoder architecture
- Transformers for Vision
  - We need to deal with $O(N^2 d)$
  - ViT: compress using pre-processing
  - DETR: object detection with Transformers

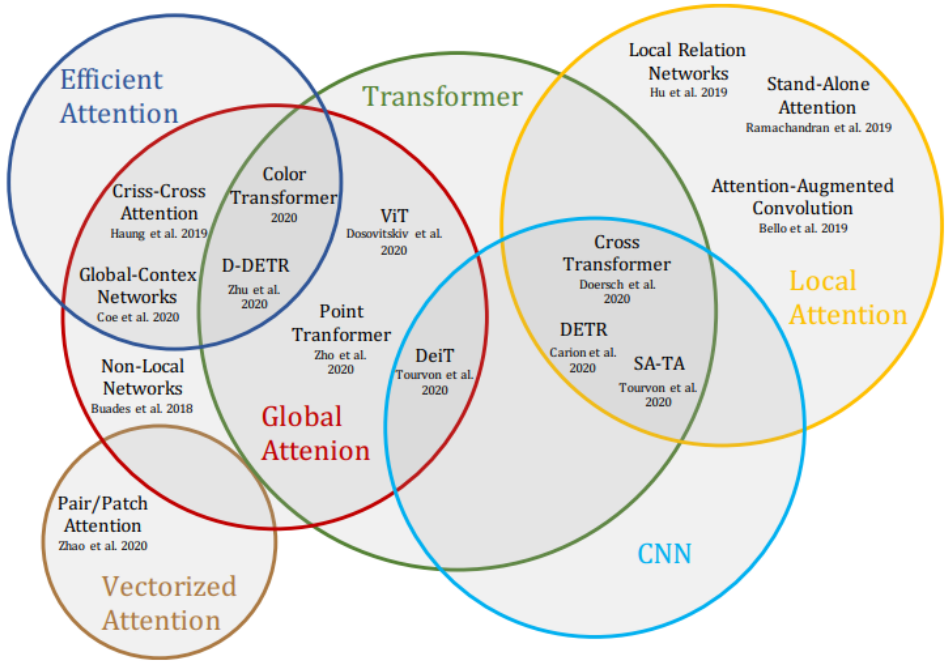


Figure from Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2021). Transformers in Vision: A Survey. arXiv preprint arXiv:2101.01169.

Conclusion

Final remarks

- There is a lot of "low-hanging fruit"
- Novel $\neq$ better
- Lots to figure out still!

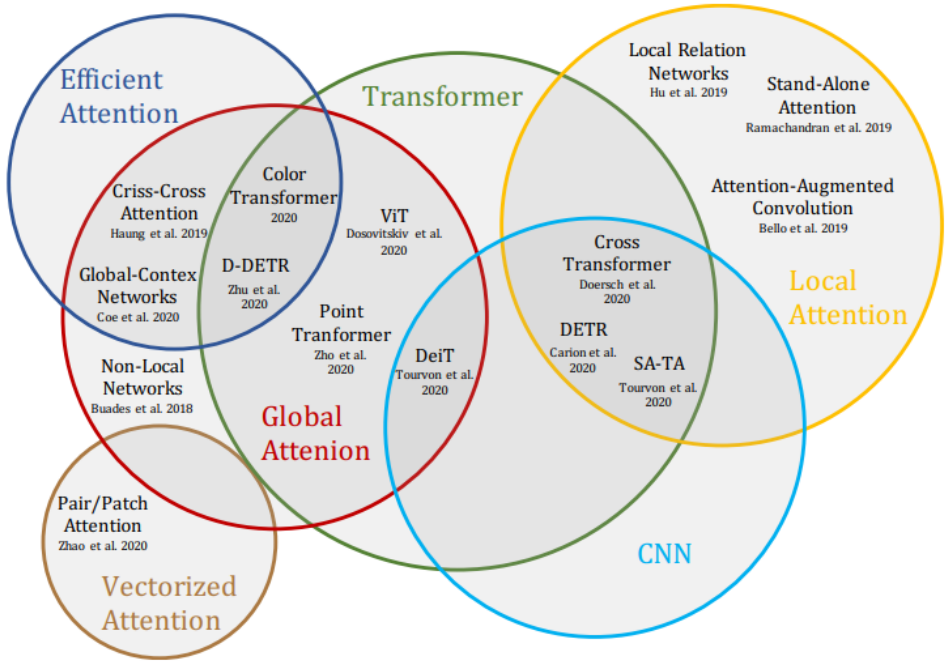


Figure from Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2021). Transformers in Vision: A Survey. arXiv preprint arXiv:2101.01169.

Conclusion

Final remarks

- There is a lot of "low-hanging fruit"
- Novel $\neq$ better
- Lots to figure out still!

Let's work on it!

https://projectforum.tudelft.nl/course_editions/13/thesis_projects/136

r.bruintjes@tudelft.nl

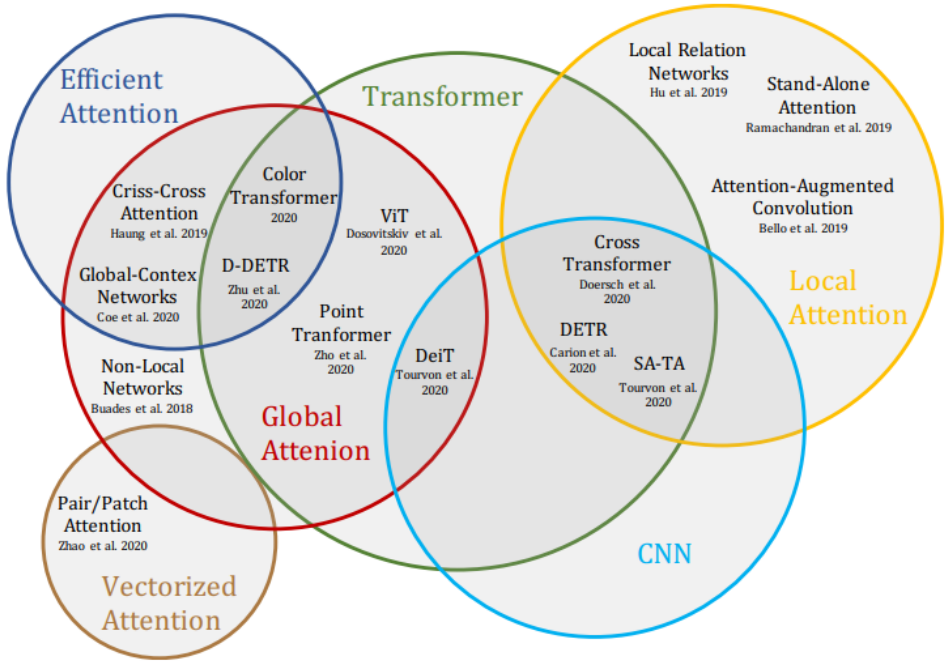


Figure from Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2021). Transformers in Vision: A Survey. arXiv preprint arXiv:2101.01169.