THE UTILITY OF TRANSFORMERS

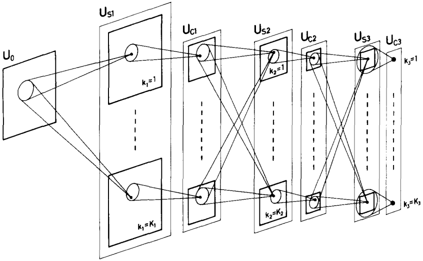
François Fleuret

UNIVERSITÉ
DE GENÈVE

Deep convolutional models

The key technology of “modern AI” are the deep convolutional models.

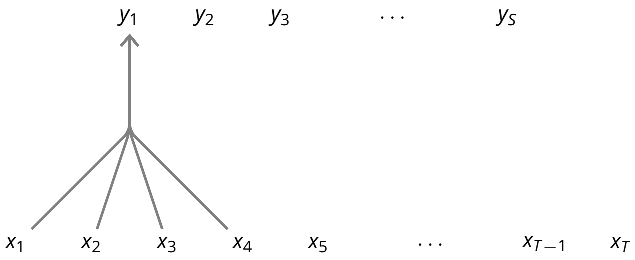
- They are powerful function approximators.
- Scale well with data set size and computation.
- Fitting for hierarchical signal structures.



Neocognitron (Fukushima, 1980)

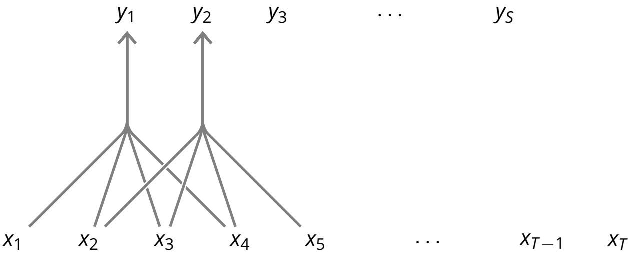
Deep convolutional models

A convolution applies the **same linear operation** at every location in the signal.



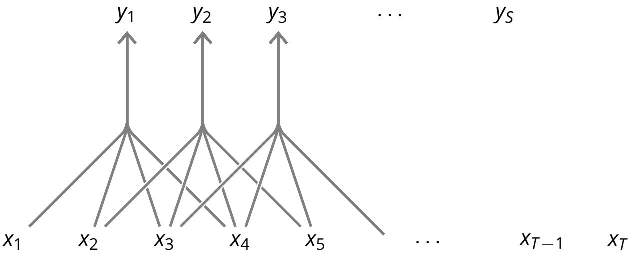
Deep convolutional models

A convolution applies the **same linear operation** at every location in the signal.



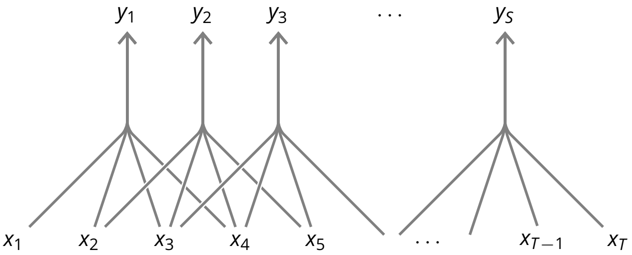
Deep convolutional models

A convolution applies the **same linear operation** at every location in the signal.



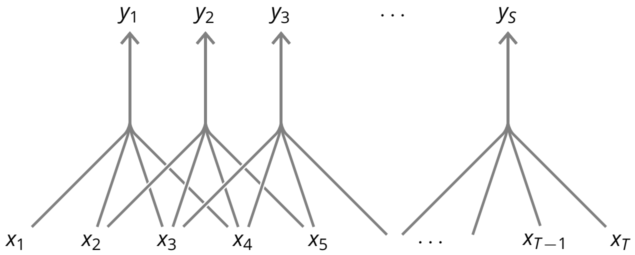
Deep convolutional models

A convolution applies the **same linear operation** at every location in the signal.



Deep convolutional models

A convolution applies the **same linear operation** at every location in the signal.



Such mechanisms are very efficient for image or sound processing where the signal is stationary and local structures are very informative.

Attention mechanisms

However some tasks involve more than hierarchical structures, e.g. translation:

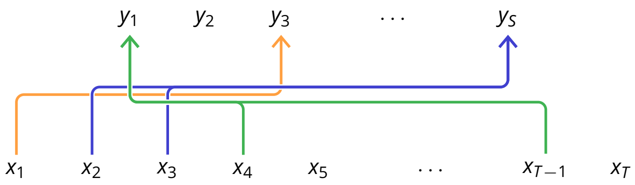
> "**An apple** that had been on the tree in the garden for weeks had finally been **picked up.**"
>
> "**Une pomme** qui était sur l'arbre du jardin depuis des semaines avait finalement été **ramassée.**"

Attention mechanisms

However some tasks involve more than hierarchical structures, e.g. translation:

> "**An apple** that had been on the tree in the garden for weeks had finally been **picked up.**"
>
> "**Une pomme** qui était sur l'arbre du jardin depuis des semaines avait finalement été **ramassée.**"

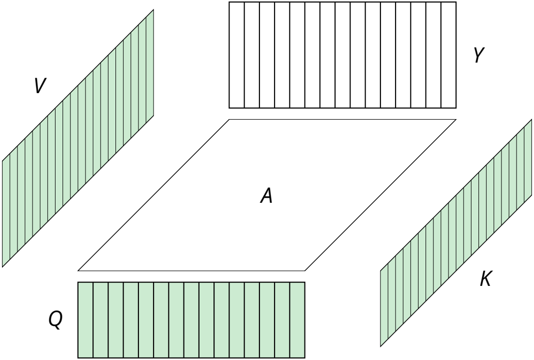
It has motivated the development of **attention-based processing** to transport information from parts of the signal to other parts dynamically identified.



Attention mechanisms

Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.

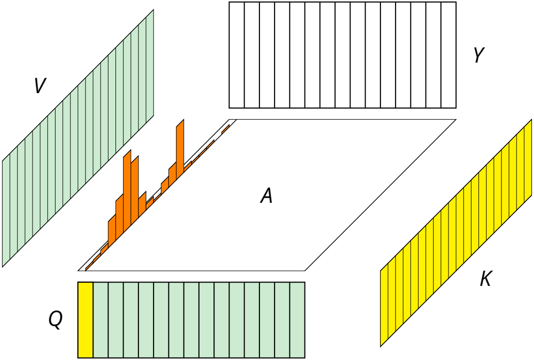
$$A = \mathrm{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \qquad Y = AV$$



Attention mechanisms

Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.

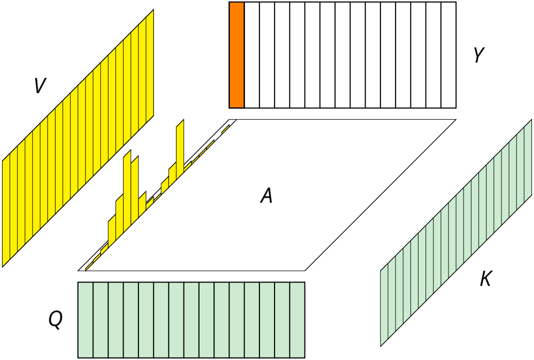
$$A = \text{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \qquad Y = AV$$



Attention mechanisms

Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.

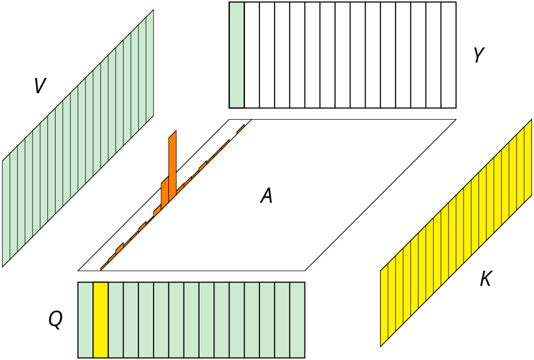
$$A = \text{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \qquad Y = AV$$



Attention mechanisms

Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.

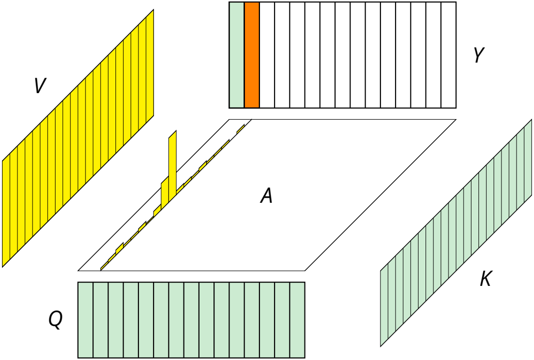
$$A = \mathrm{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \qquad Y = AV$$



Attention mechanisms

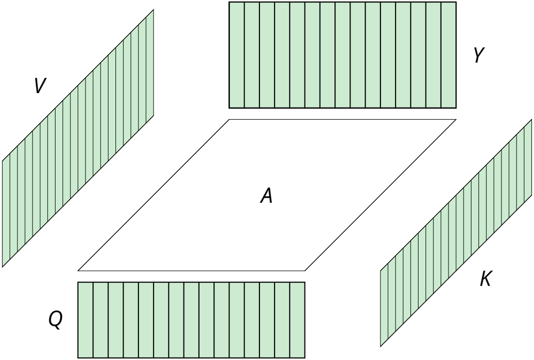
Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.

$$A = \mathrm{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \qquad Y = AV$$

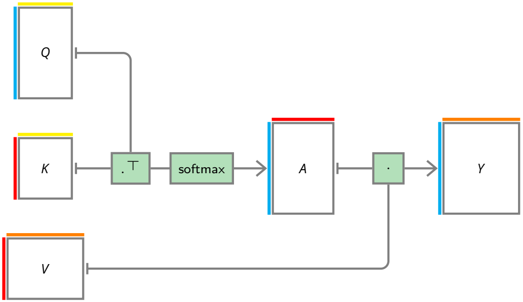


Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.

$$A = \text{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \qquad Y = AV$$



Attention mechanisms



$$A = \text{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right)$$

$$Y = AV.$$

Single-head attention operation

Attention mechanisms

A standard attention layer takes as input two sequences X and X' and computes

$$
\begin{aligned}
K &= X{W^K}^\top \\
V &= X'{W^V}^\top \\
Q &= X'{W^Q}^\top \\
A &= \mathrm{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right) \\
Y &= AV
\end{aligned}
$$

Attention mechanisms

A standard attention layer takes as input two sequences X and X' and computes

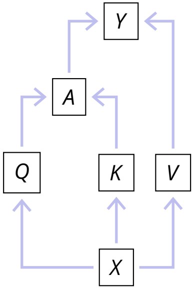
$$K = X{W^K}^\top$$

$$V = X'{W^V}^\top$$

$$Q = X'{W^Q}^\top$$

$$A = \text{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right)$$

$$Y = AV$$



When $X = X'$, this is **self attention**

Attention mechanisms

A standard attention layer takes as input two sequences X and X' and computes

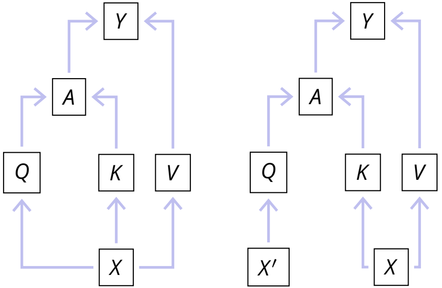
$$K = X{W^K}^\top$$

$$V = X'{W^V}^\top$$

$$Q = X'{W^Q}^\top$$

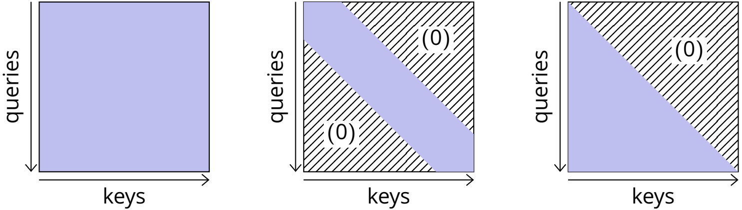
$$A = \text{softmax}_{row}\left(\frac{QK^\top}{\sqrt{d}}\right)$$

$$Y = AV$$



When $X = X'$, this is **self attention**, otherwise **cross attention.**

It may be useful to mask the attention matrix, for instance in the case of self-attention, for computational reasons, or to make the model causal for auto-regression.



Full attention

Local attention

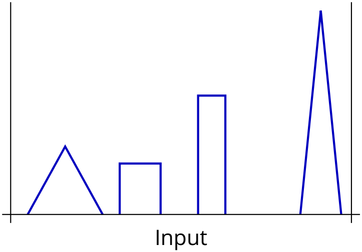
$|i - j| > \Delta \Rightarrow A_{i,j} = 0$

Causal attention

$j > i \Rightarrow A_{i,j} = 0$

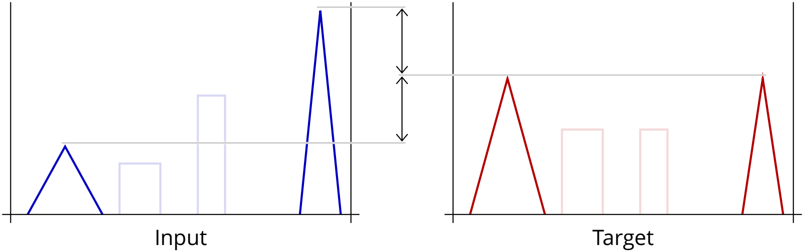
Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.

Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.



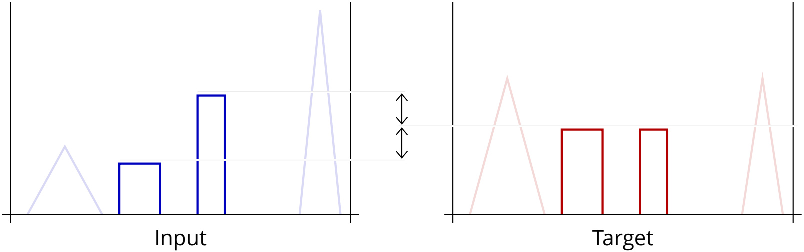
Toy seq2seq example

Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.



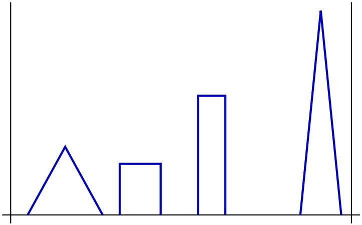
Toy seq2seq example

Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.

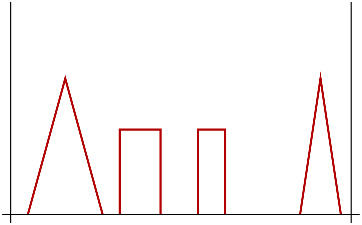


Toy seq2seq example

Consider a task with 1d sequences composed of two triangles and two rectangles, where the goal is to average heights in each **pair of shapes**.

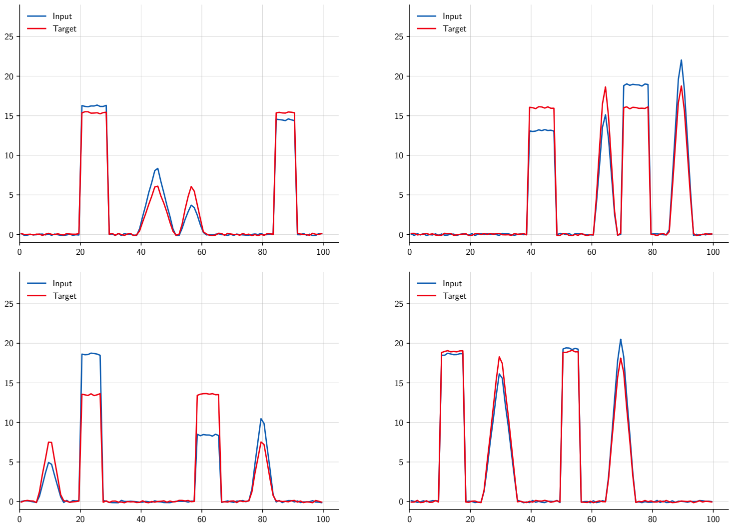


Input



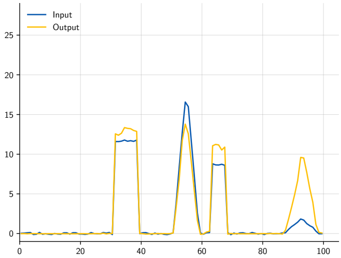
Target

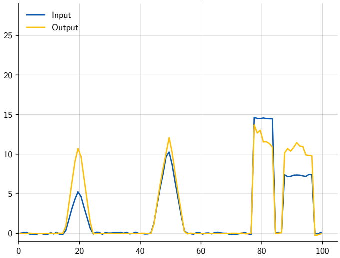
Toy seq2seq example



Toy seq2seq example

```
Sequential(
  (0): Conv1d(1, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (1): ReLU()
  (2): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (3): ReLU()
  (4): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (5): ReLU()
  (6): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (7): ReLU()
  (8): Conv1d(64, 1, kernel_size=(5,), stride=(1,), padding=(2,))
)
```





Toy seq2seq example

The poor performance of this model is not surprising given its inability to channel information from “far away” in the signal.

More layers, global averaging, or fully connected layers could possibly solve the problem. However it is more natural to equip the model with the ability to fetch information from parts of the signal that it actively identifies as relevant.

This is exactly what an **attention layer** does.

Toy seq2seq example

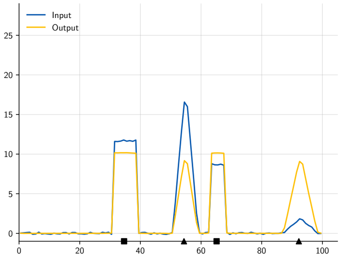
```
class SelfAttentionLayer(nn.Module):
    def __init__(self, in_dim, out_dim, key_dim):
        super().__init__()
        self.conv_Q = nn.Conv1d(in_dim, key_dim, kernel_size = 1, bias = False)
        self.conv_K = nn.Conv1d(in_dim, key_dim, kernel_size = 1, bias = False)
        self.conv_V = nn.Conv1d(in_dim, out_dim, kernel_size = 1, bias = False)

    def forward(self, x):
        Q = self.conv_Q(x)
        K = self.conv_K(x)
        V = self.conv_V(x)
        A = torch.einsum('nct,ncs->nts', Q, K).softmax(2)
        y = torch.einsum('nts,ncs->nct', A, V)
        return y
```

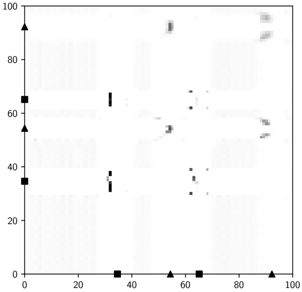
Toy seq2seq example

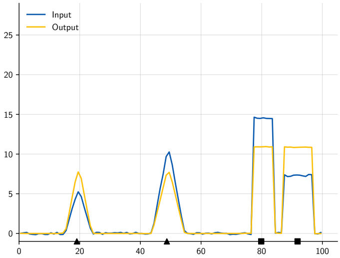
```
class SelfAttentionLayer(nn.Module):
    def __init__(self, in_dim, out_dim, key_dim):
        super().__init__()
        self.conv_Q = nn.Conv1d(in_dim, key_dim, kernel_size = 1, bias = False)
        self.conv_K = nn.Conv1d(in_dim, key_dim, kernel_size = 1, bias = False)
        self.conv_V = nn.Conv1d(in_dim, out_dim, kernel_size = 1, bias = False)

    def forward(self, x):
        Q = self.conv_Q(x)
        K = self.conv_K(x)
        V = self.conv_V(x)
        A = torch.einsum('nct,ncs->nts', Q, K).softmax(2)
        y = torch.einsum('nts,ncs->nct', A, V)
        return y
```

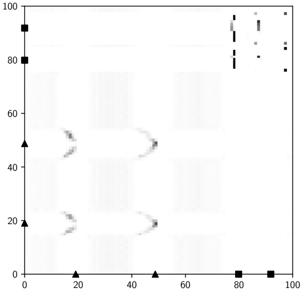
```
Sequential(
  (0): Conv1d(1, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (1): ReLU()
  (2): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (3): ReLU()
  (4): SelfAttentionLayer(in_channels=64, out_channels=64, key_channels=64)
  (5): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (6): ReLU()
  (7): Conv1d(64, 1, kernel_size=(5,), stride=(1,), padding=(2,))
)
```

Toy seq2seq example

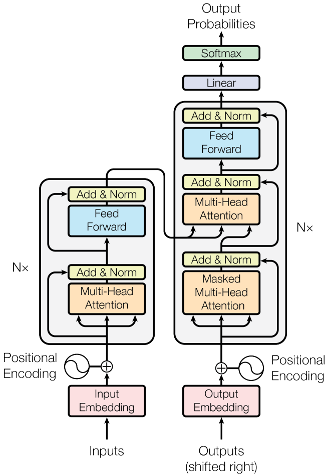






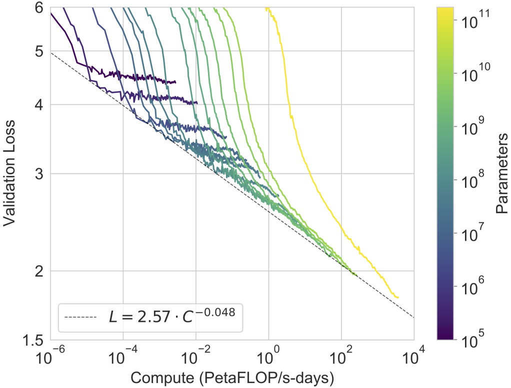


The standard transformer model combines a stack of self-attention layers in an encoder, and a stack of self-attention and cross-attention layers in a decoder.



(Vaswani et al., 2017)

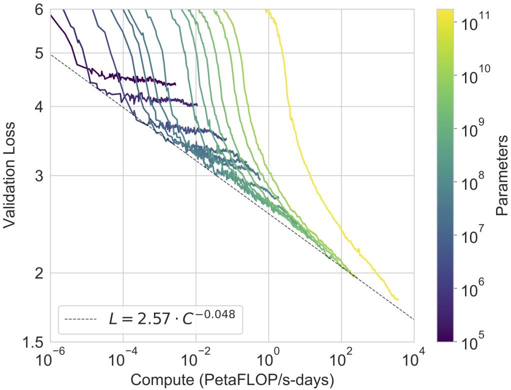
Transformers exhibit extremely good transfer capabilities and scale well.



(Brown et al., 2020)

Transformers

Transformers exhibit extremely good transfer capabilities and scale well.



(Brown et al., 2020)

All SoTA methods across NLP tasks are transformer-based, and these models are getting on par or SoTA in computer vision.

Large language models have been shown to exhibit some “zero shot learning” capabilities when they are properly “primed” (Brown et al., 2020).

For instance using HuggingFace’s `gpt2-xl` model with 1.6B parameters, we can get these sentence completions, where the priming text is between <>:

```
<Cherry is red, lettuce is green, lemon is> yellow, and orange is blue.

<Cherry is sweet, lettuce is bland, lemon is> sour, and orange is bitter.

<Cherry is a fruit, lettuce is a vegetable, lemon is> a fruit, and so on.
```

Transformers

Explaining a Joke

I will explain these jokes:
(1) The problem with kleptomaniacs is that they always take things literally.
Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't fully understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.
(2) Always borrow money from a pessimist. They'll never expect it back.
Explanation: Most people expect you to pay them back when you borrow money, however a pessimist is someone who always assumes the worst, so if you borrow money from them, they will expect that you won't pay them back anyways.

Input: I was supposed to start writing the paper at 5:00 PM. But then I started playing with this cool new language model for 10 minutes. 10 minutes later, it's suddenly 9:30 PM!
Model Output: This joke is about how time flies when you're having fun. The person was supposed to start writing a paper at 5:00 PM, but they started playing with a language model instead. They thought they were only playing with it for 10 minutes, but it was actually 4.5 hours.

Input: I think I understand why me and my partner are such a good match. I get excited when I smell food and become super alert when the doorbell rings. My girlfriend Shelley becomes ecstatic when she sees a person who she just saw 5 hours ago.
Model Output: This joke is about dogs. Dogs get excited when they smell food and when the doorbell rings. The girlfriend is excited to see someone she just saw 5 hours ago, which is a trait of dogs.

Input: I was going to fly to visit my family on April 6th. My mom said "Oh great, your step dad's poetry reading is that night!" So now I'm flying in on April 7th.
Model Output: The joke is that the speaker's mother is trying to get them to go to their step dad's poetry reading, but the speaker doesn't want to go, so they are changing their flight to the day after the poetry reading.

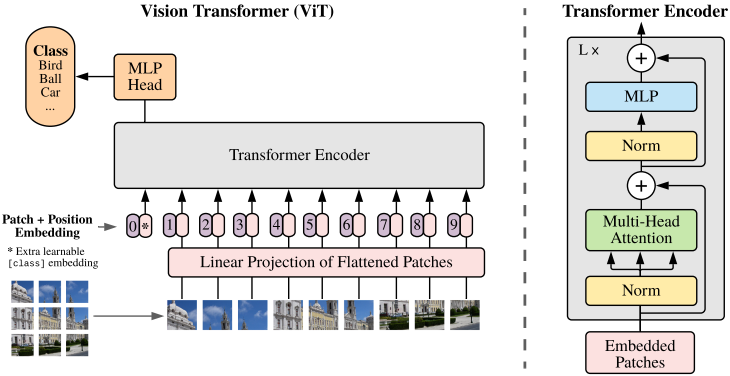
Input: I tried 10,000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.
Model Output: This joke is a pun. A neural network is a computer program that can learn from data. A "seed" is a number that is used to initialize a random number generator. A "good seed" is a number that produces a good random number generator. The phrase "no good deed goes unpunished" means that if you do something good, you will be punished for it.

Input: Reading "War and Peace" doesn't scare me. I once had to review the CL for my intern's whole project on his final day!
Model Output: 'War and Peace' is a very long book, so reading it is a daunting task. However, the speaker is saying they had to review the code for their intern's project on their final day, which is even more daunting.

Input: What's the difference between a zebra and an umbrella? One is a striped animal related to horses, another is a device you use to stop the rain from falling on you.
Model Output: This joke is an anti-joke. The joke is that the answer is obvious, and the joke is that you were expecting a funny answer.

(Chowdhery et al., 2022)

Transformers



(Dosovitskiy et al., 2020)

PICOCLVR

UNIVERSITÉ DE GENÈVE

PicoCLVR

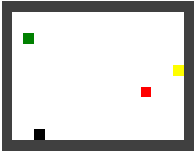
The PicoCLVR is a toy task designed to assess the ability of an attention-based auto-regressive model to generate an image composed of elements whose positions are constrained by a series of NLP statements.

Each sample is generated by creating a 12×16 image with up to 5 colored pixels drawn at random locations, and then by sampling a few Boolean properties regarding their placement.

Such a sample is encoded as a sequence of tokens for the properties first, separated by a specific token, followed by the image encoded as a sequence of pixels in raster-scan order.

PicoCLVR

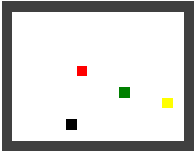
```
yellow right of red <sep> there is green <sep> black below red <sep> green
above yellow <sep> green left of red <sep> black left of red <sep> green left
of black <sep> black left of yellow <img> white white white white white white
white white white white white white white white white white white white white
white white white white white white white white white white white white white
white green white white white white white white white white white white white
white white white white white white white white white white white white white
white white white white white white white white white white white white white
white white white white white white white white white white white white white
white white white white white white white white white white white yellow white
white white white white white white white white white white white white white
white white white white white white white white white white white white white
white red white white white white white white white white white white white
white white white white white white white white white white white white white
white white white white white white white white white white white white white
white white white white white white white white white white white white white
white white white black white white white white white white white white white
white white white white
```



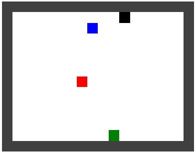
PicoCLVR

Training examples.

```
black below yellow <sep> black below green <sep> yellow right of green <sep>
yellow right of red <sep> red left of yellow <sep> yellow above black <sep>
green left of yellow <sep> yellow below green
```

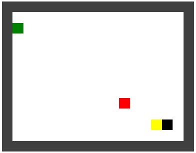


```
green below red <sep> black right of green <sep> red left of black
```

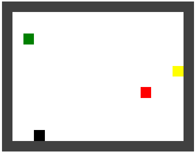


Training examples.

`yellow right of red <sep> there is red`



`yellow right of red <sep> there is green <sep> black below red <sep> green above yellow <sep> green left of red <sep> black left of red <sep> green left of black <sep> black left of yellow`

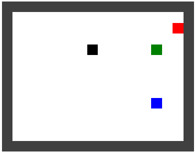


PicoCLVR

Training examples.

`blue left <sep> blue top <sep> there is blue`

`blue bottom <sep> there is black <sep> blue below green <sep> red right of green <sep> blue below red <sep> red top`



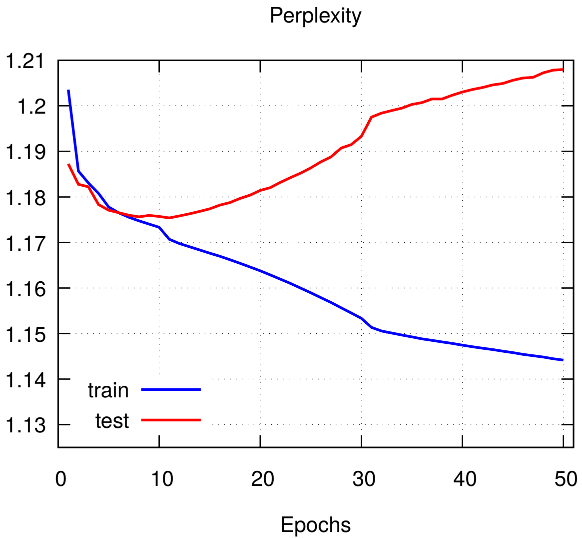
We use a standard causal transformer encoder with the following parameterization (38M parameters):

```
dim_model 512
dim_keys 64
dim_hidden 2048
nb_heads 8
nb_blocks 12
dropout 0.1
```

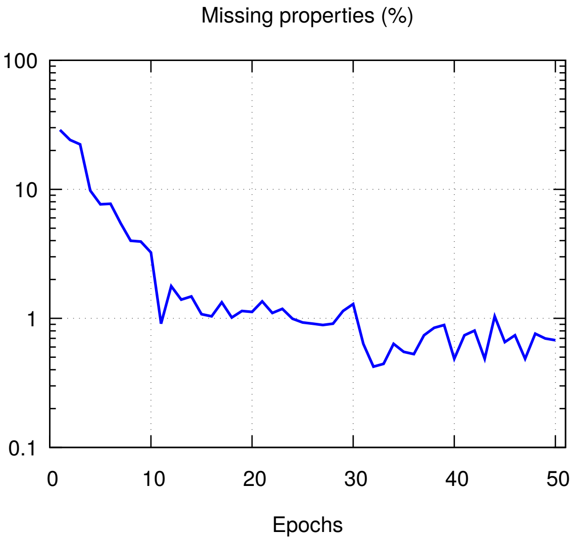
Training is done with 250k samples and the following setup:

```
nb_epochs 50
batch_size 25
optim adam
learning_rate 0.0001
learning_rate_schedule 10: 2e-5,30: 4e-6
```

PicoCLVR


Perplexity
1.21
1.2
1.19
1.18
1.17
1.16
1.15
1.14
1.13
0
10
20
30
40
50
Epochs
train
test

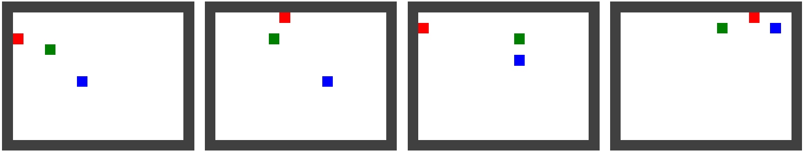
PicoCLVR



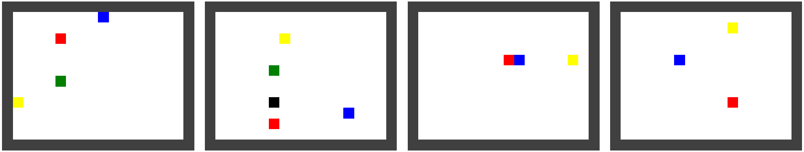
PicoCLVR

Test examples.

`red above green <sep> green top <sep> blue right of red`

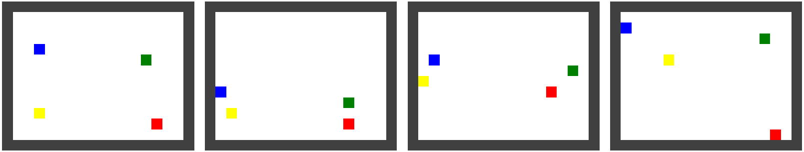


`there is red <sep> there is yellow <sep> there is blue`

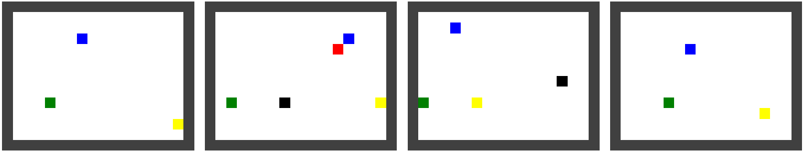


Test examples.

`red below yellow <sep> yellow below green <sep> green below blue <sep> red right <sep> yellow left <sep> green right <sep> blue left`



`green bottom <sep> yellow bottom <sep> green left of blue <sep> yellow right of blue <sep> blue top`



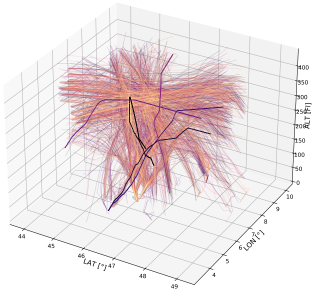
WIND PREDICTION ON AIRPLANE TRAJECTORIES

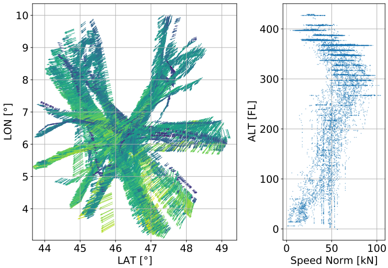
Wind prediction along airplane trajectories

- Wind conditions are very important for air traffic control.
- Measurements are available only a 2-3 times per days.
- Controllers often infer the wind conditions from the aircrafts behavior.
- Aircrafts broadcast every 4s their position, pressure, and air speed.

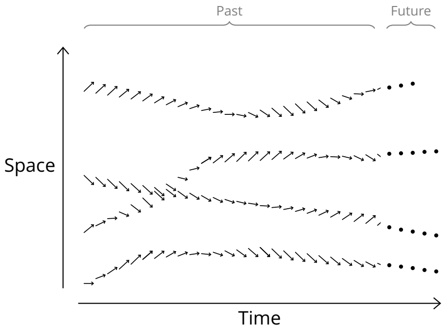
Wind prediction along airplane trajectories

- Wind conditions are very important for air traffic control.
- Measurements are available only a 2-3 times per days.
- Controllers often infer the wind conditions from the aircrafts behavior.
- Aircrafts broadcast every 4s their position, pressure, and air speed.

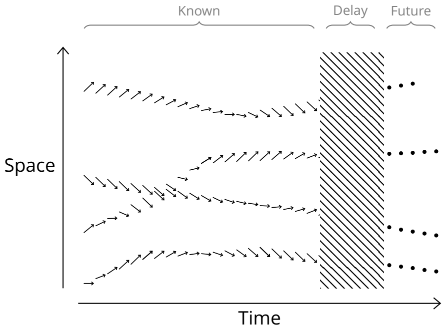




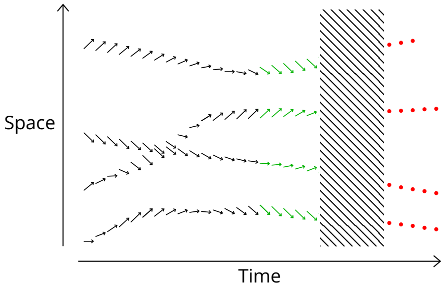
Wind prediction along airplane trajectories

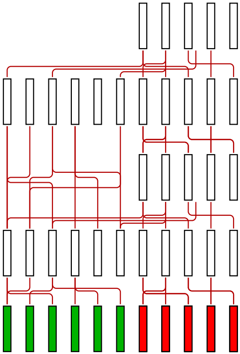


Wind prediction along airplane trajectories



Wind prediction along airplane trajectories





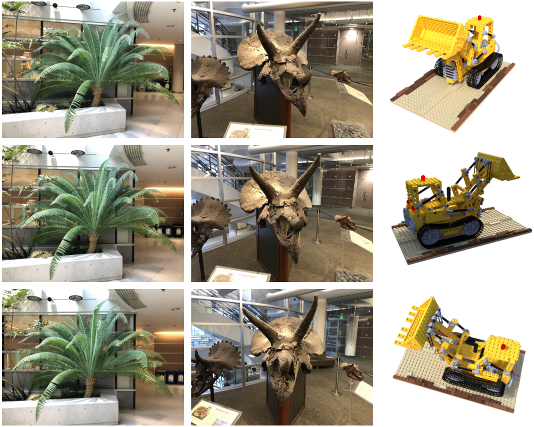
Wind prediction along airplane trajectories

Method	MSE
k-NN	10.19
GKA	10.38
GKA + MLP	9.65
Transformer	9.12

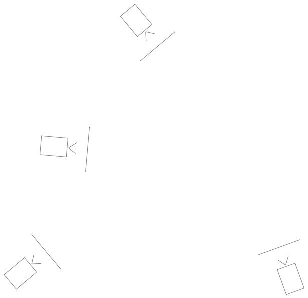
GEOMETRIC RADIANCE FIELD MODELING

GeoNerf

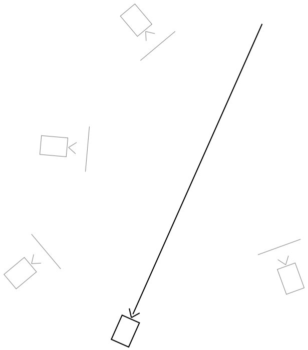
- Given a series of images of a scene from different angles with their camera calibration, build a view from a novel position.
- Deal with specularities and reflections by modeling the radiance field.



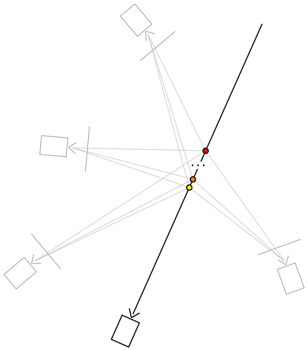
GeoNerf



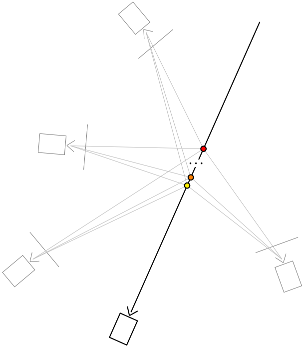
GeoNerf

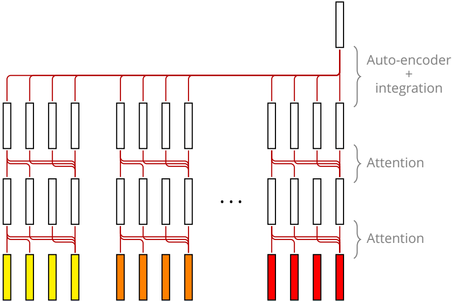


GeoNerf



GeoNerf





GeoNerf

(video)

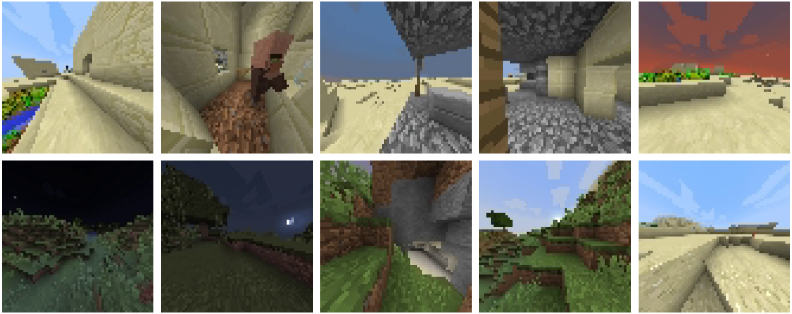
(video)

(video)

IMITATION LEARNING IN MINECRAFT

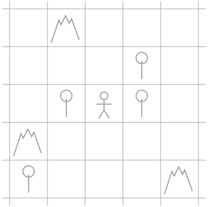
MineRL Agent

- Image-based, first-person 3d perspective.
- Rich environment, short-term navigation constraints and trajectory control.
- Complex multi sub-tasked long-term planning.
- Learning a policy by imitation, from hundreds of recorded games.

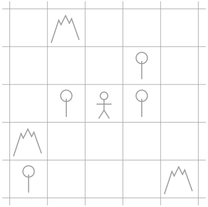


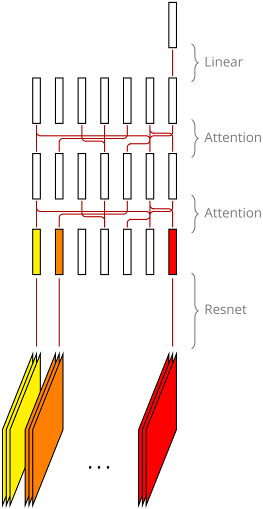
(video)

MineRL Agent



MineRL Agent





MineRL Agent

(video)

Conclusion

- Transformers works well in many application domains.
- They scale very well.
- They can be pre-trained / fine-tuned.
- They are likely here to stay.

Questions?

References

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. **Language models are few-shot learners**. *CoRR*, abs/2005.14165, 2020.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. Dai, T. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. **Palm: Scaling language modeling with pathways**. *CoRR*, abs/2204.02311, 2022.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. **An image is worth 16x16 words: Transformers for image recognition at scale**. *CoRR*, abs/2010.11929, 2020.

K. Fukushima. **Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position**. *Biological Cybernetics*, 36(4):193–202, April 1980.

M. Johari, Y. Lepoittevin, and F. Fleuret. **GeoNeRF: generalizing nerf with geometry priors**. In *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. To appear.

A. Kanervisto, S. Milani, K. Ramanauskas, N. Topin, Z. Lin, J. Li, J. Shi, D. Ye, Q. Fu, W. Yang, W. Hong, Z. Huang, H. Chen, G. Zeng, Y. Lin, V. Micheli, E. Alonso, F. Fleuret, A. Nikulin, Y. Belousov, O. Svidchenko, and A. Shpilman. **MineRL Diamond 2021 Competition: Overview, results, and lessons learned**. *CoRR*, abs/2202.10583, 2022.

A. Pannatier, R. Picatoste, and F. Fleuret. **Efficient wind speed nowcasting with GPU-accelerated nearest neighbors algorithm**. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2022. To appear.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. **Attention is all you need**. *CoRR*, abs/1706.03762, 2017.