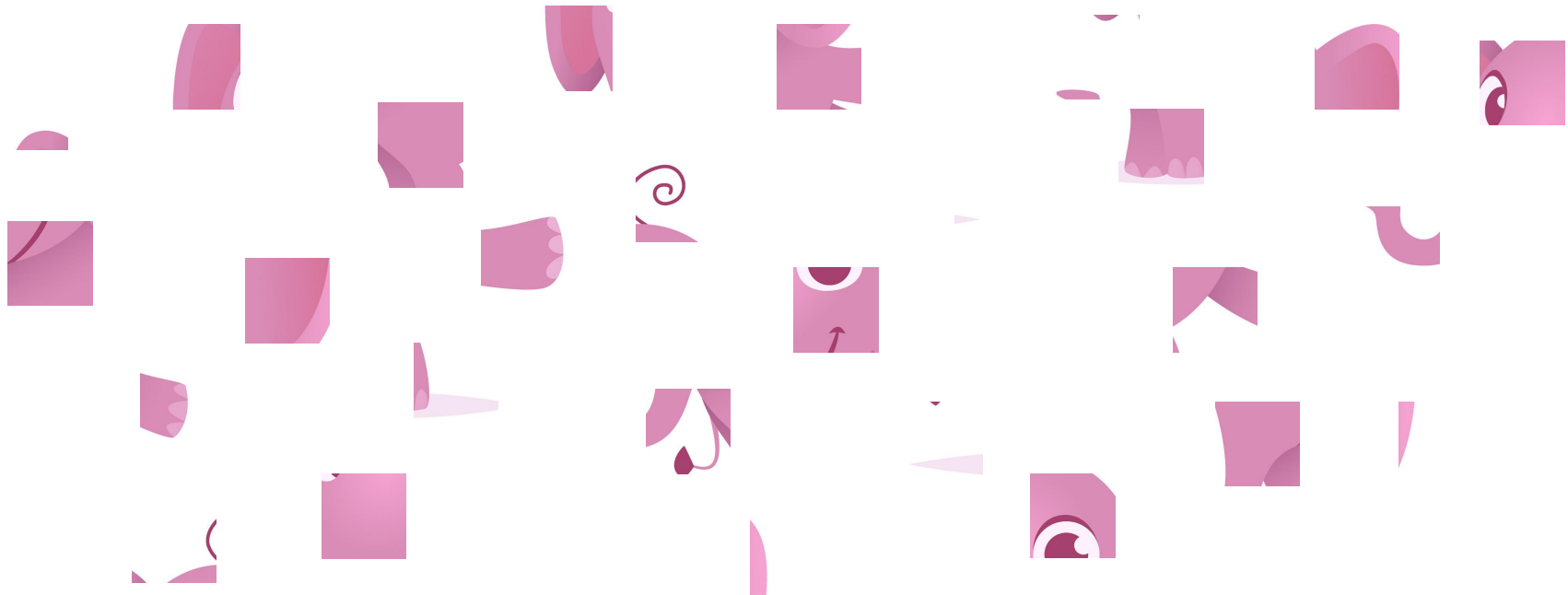


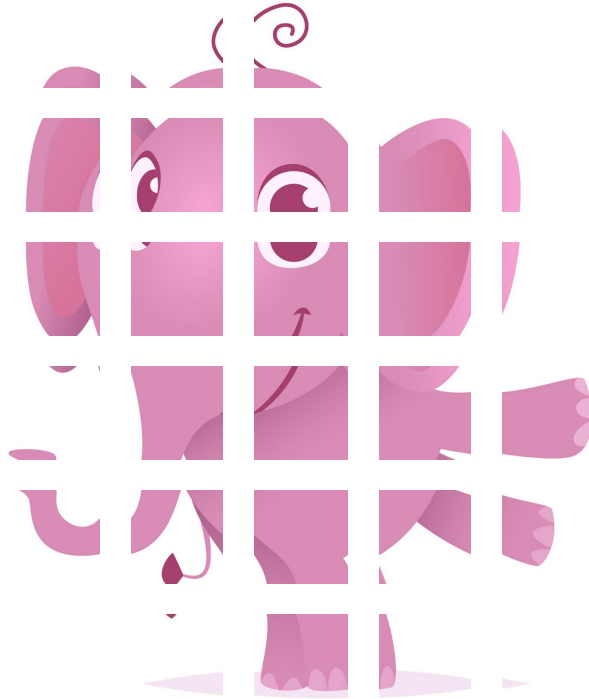
Can you recognize it?



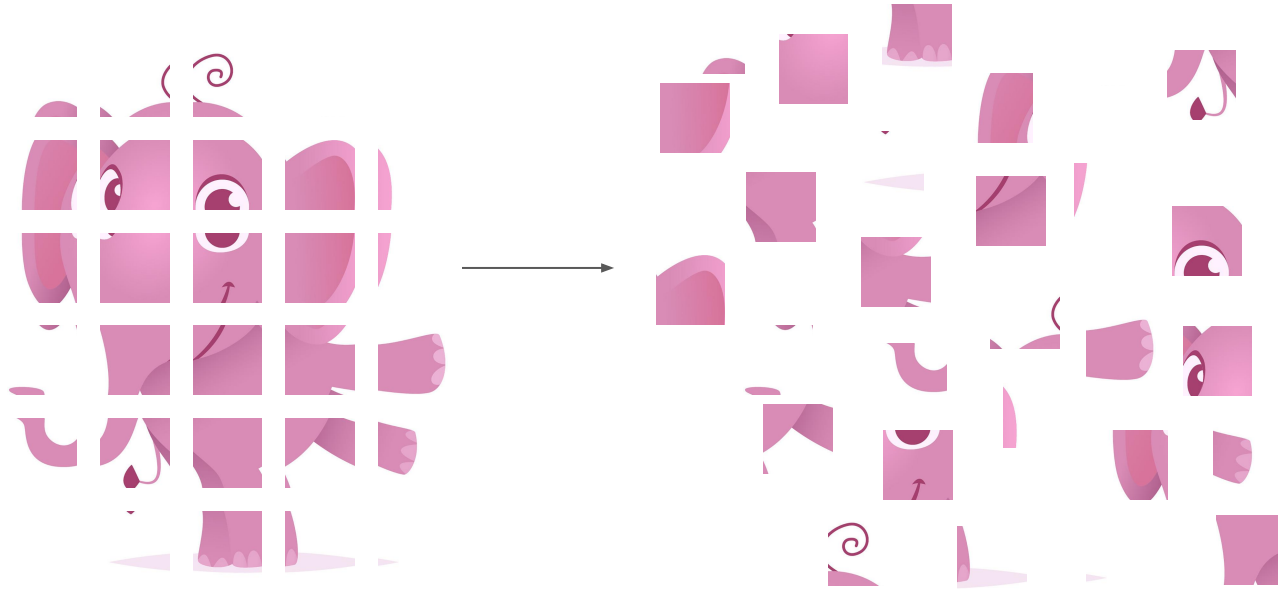
What about now?



A Happy Pink Elephant



Attention models interpret inputs as



Input

What attention models see

Just a set of
unordered pixels

Who am I?

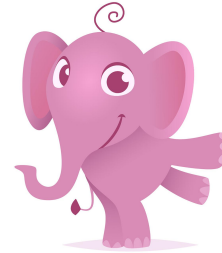


Meena Alfons

MSc. Artificial Intelligence
2022 at **VU** 

Software Engineer since 2013
At  **MessageBird** since late 2019

This is my masters thesis.



What is this about?

Thesis Title:

“Multi-dimensional Positional **Self-attention** With Differentiable Span”

Main contributions:

- SIREN-based Positional Encoding
- Differentiable attention span

Why do we care about self-attention?

Natural Language Processing

- Machine Translation
- Question Answering
- Sentiment Analysis
- Summarization
- Pre-trained language models (BERT, GPT-3)

Computer Vision

- Image Classification
- Image Generation
- Object Detection

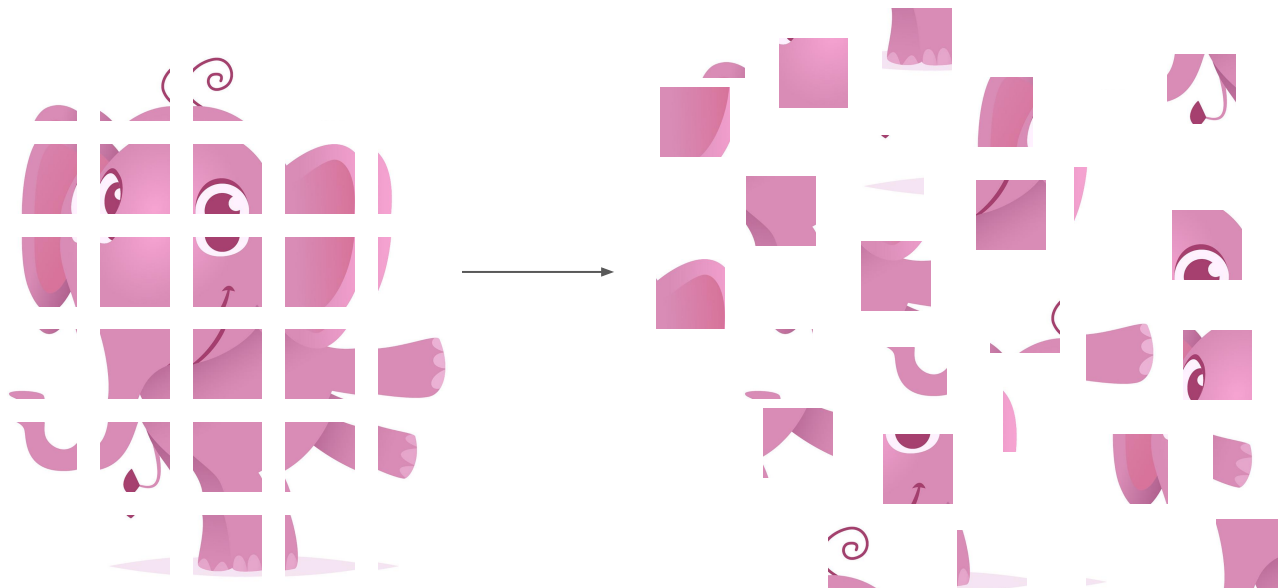
Multi-modal tasks

- Multimedia (image, video) Description
- Speech Recognition

Recommender Systems

Graph-based Systems

Back to the happy pink elephant



Input

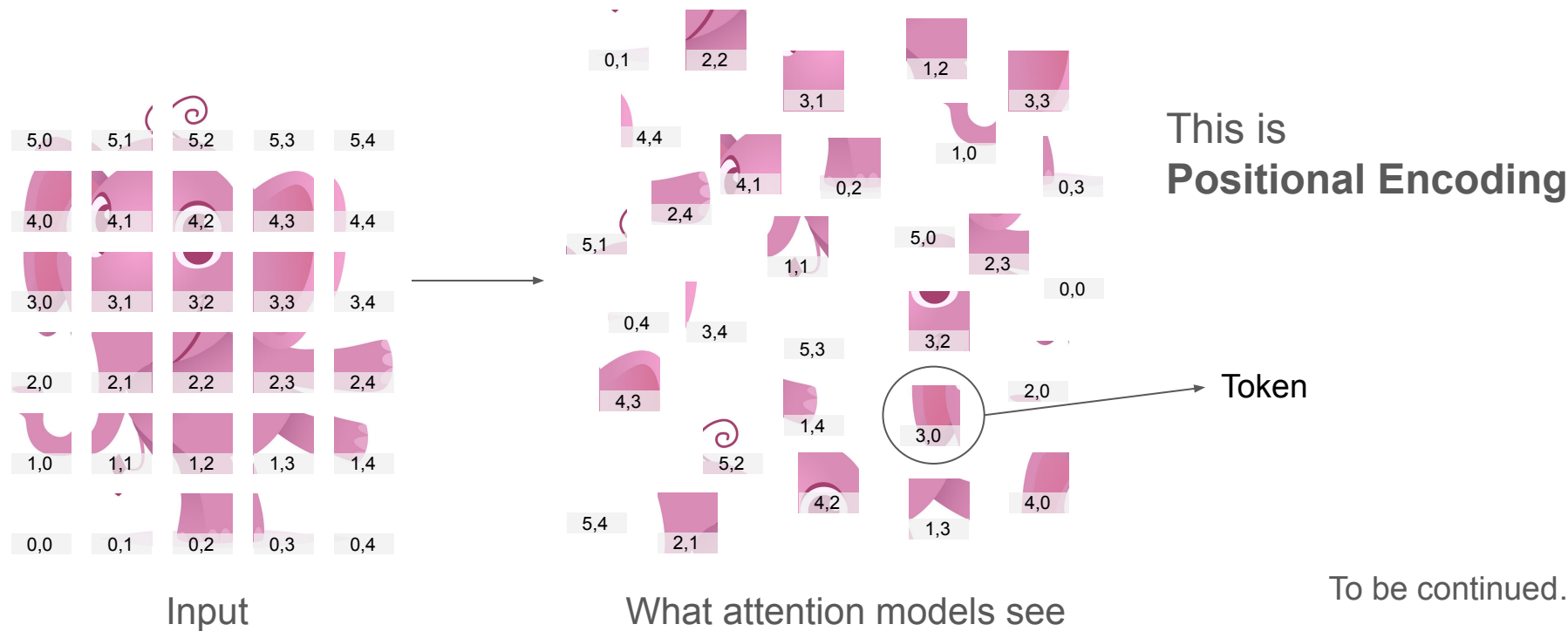
What attention models see

Input consists of:

- **Colors**
- Their **positions**

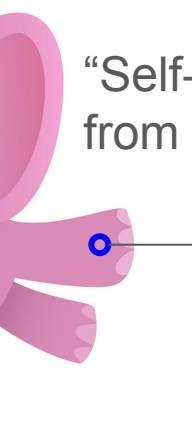
How to help attention models recognize the position?

Solution: Add position information to each pixel



What is Self-attention?

“Self-attention is a way for each pixel to **collect more information** about itself from other pixels around it.”



→ I'm a pink pixel

↓ 1st time

I'm pink and I'm part of a wide pink area

↓ 2nd time

I'm part of a thick pink leg

↓ 3rd time

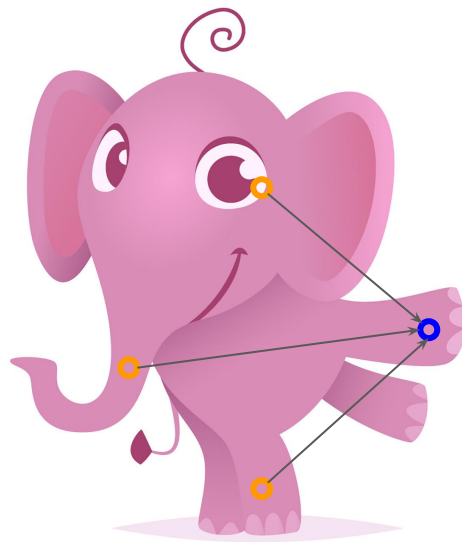
I'm part of a pink elephant

↓ 4th time

I'm part of a pink elephant on a white background

Applying self-attention
multiple times

How self-attention works?



How important is another pixel (Key) for this pixel (Query)?

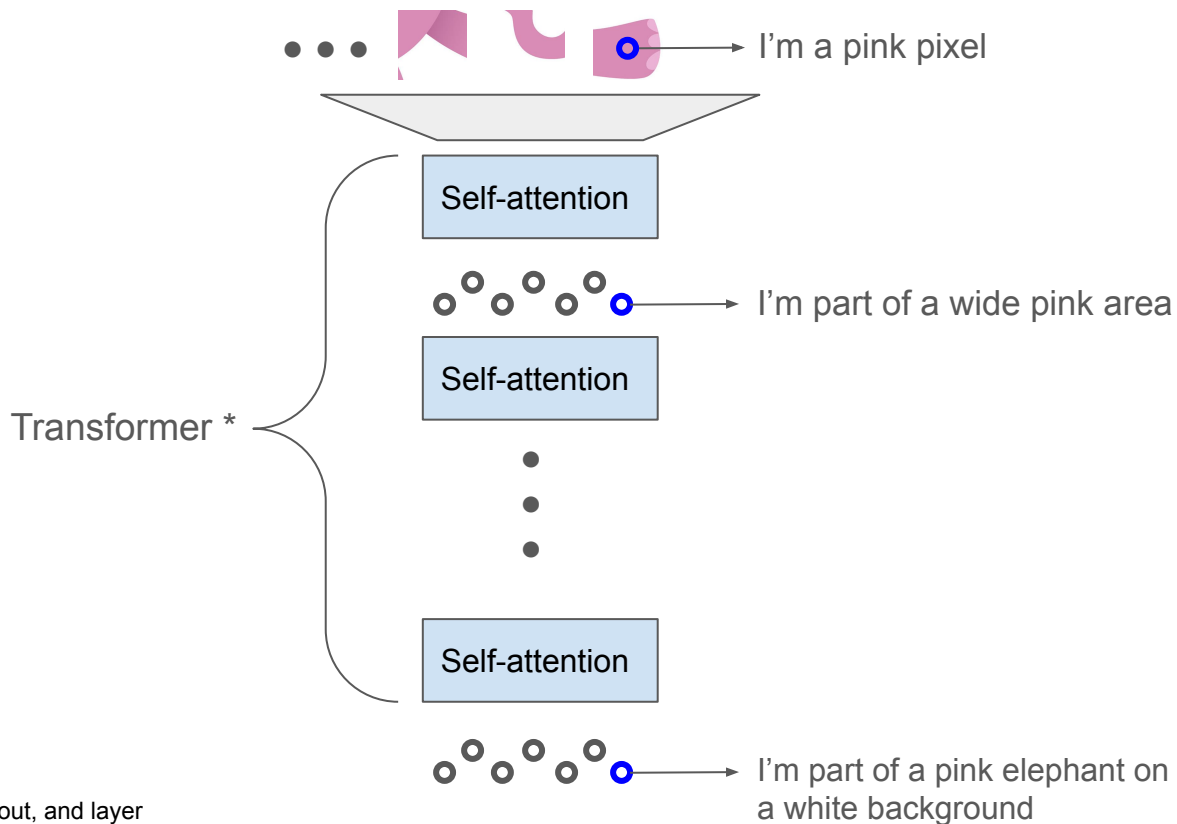
Attention scores

● Query

● Keys

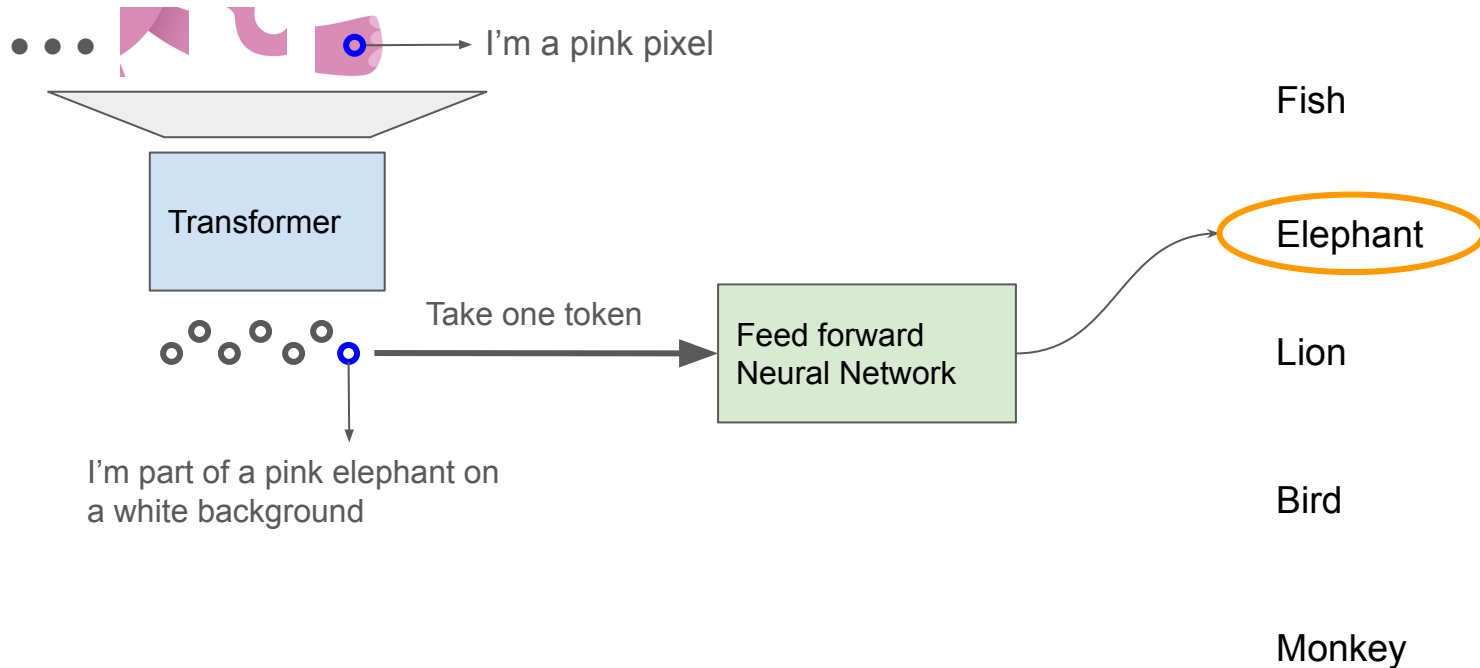
The influence of a Key on a Query depends on the **attention score**

The Transformer



* Residual connections, dropout, and layer normalization are omitted

Transformer-based classifier



Now we know

- What self-attention does
- How to use it for classification tasks
- The importance of positional encoding for self-attention

Let's get back to **Positional Encoding**

Positional Encoding Landscape

| Factors | Previous Work | Baseline | Our Work |
|----------------------|---------------|-----------------------------------|-----------------------------|
| absolute/relative | ... | relative | relative |
| representation | ... | fixed sin features + linear layer | normalised position + SIREN |
| learnable parameters | ... | $O(1)$ | $O(1)$ |
| discrete/continuous | ... | continuous | continuous |
| and more... | ... | ... | ... |

↑
Based on
Transformer-XL

↑
SIREN-based
positional encoding

Why SIREN-based positional encoding?

SIREN = sinusoidal representation network

A multi-layer perceptron with sine activation functions
(instead of tanh, ReLU, etc)

- Have universal function approximation properties
- Have superior performance in reconstruction tasks

Should work very well to capture
complex Positional Encoding



Representation: Baseline vs SIREN-based

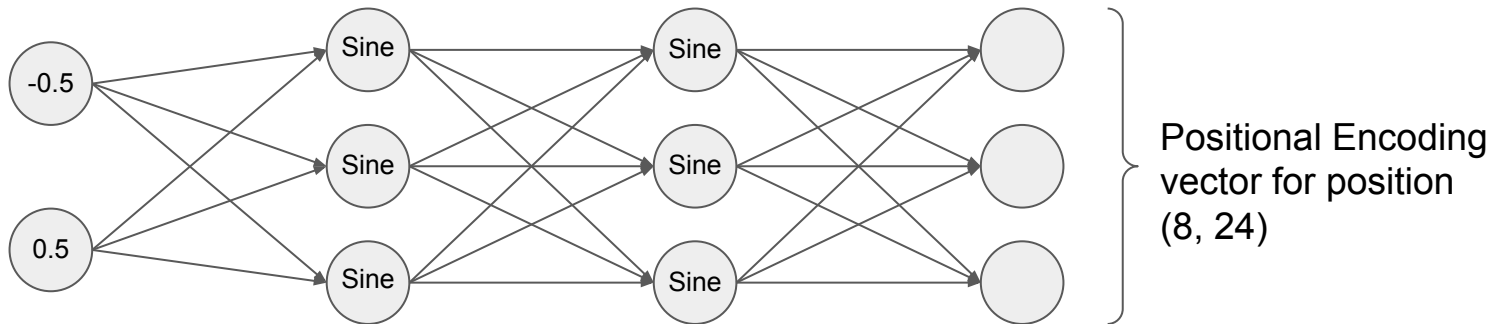
| Baseline Fixed sin features + linear layer | SIREN-based Normalized position + SIREN |
|---|--|
| <u>Predefined frequencies</u> | <u>Learned frequencies</u> at each layer |
| Linear combination | Multi-layer network |
| Designed for 1D | Naturally extends to nD |
| Can be used for nD by <ul style="list-style-type: none">• Addition: constrained$f(\vec{x}) = f_1(x_1) + f_2(x_2) + \dots$• Concatenation: use less features per dimension | Can model complex functions in nD Approximates $f(\vec{x})$ |

How?

Input size 32x32 (image)

Position: (8, 24)

Normalized position in $[-1,1]$: $(8 \times \frac{2}{32} - 1, 24 \times \frac{2}{32} - 1) = (-0.5, 0.5)$



Experiment - Baseline vs SIREN-based

Task: Image Classification

Dataset: CIFAR10 (10 classes), sequential = 1D

Repeated: 5 times for 5 different random states

Hypothesis:

- H0: No difference in validation accuracy
- H1: SIREN-based achieves higher validation accuracy than the Baseline

Results - Baseline vs SIREN-based

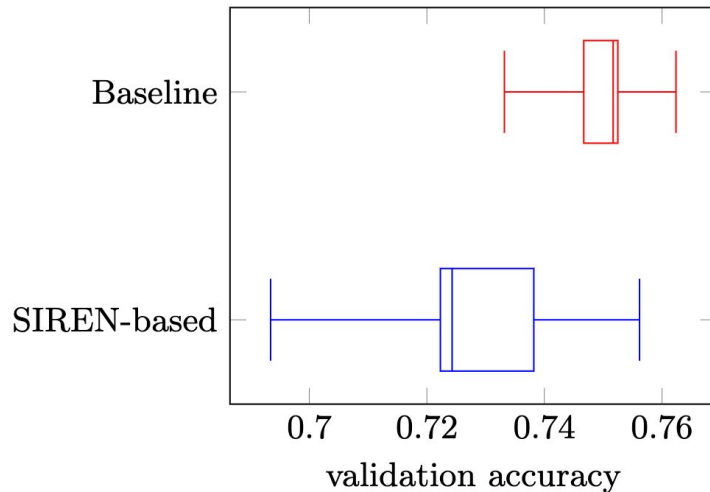
Comparable performance: max acc of **0.756**

SIREN-based is **lower** than Baseline by 2% (on avg)

- Hyperparameter sensitivity
- Why doesn't SIREN-based outperform the Baseline while it can theoretically model the Baseline? (future work)

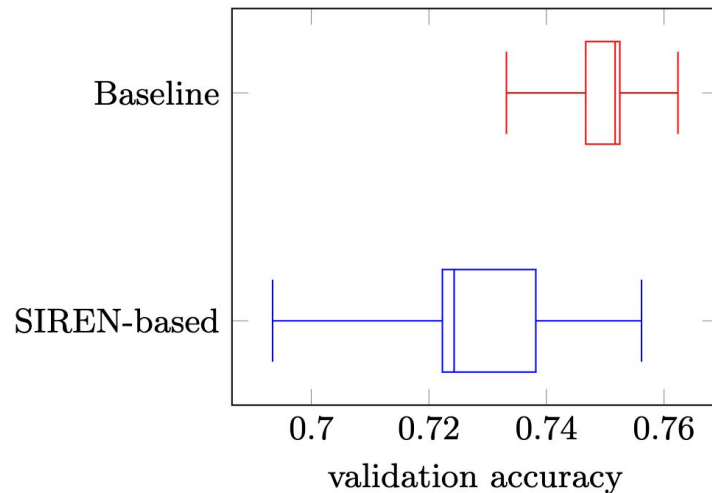
Doesn't discredit the approach

+ naturally extends to multiple dimensions



Future work - SIREN-based positional encoding

- Why doesn't it outperform the Baseline?
Comprehensive comparison is needed.
- Hyperparameter sensitivity: a more stable parameterisation is needed



What was covered?

- ✓ Self-attention
- ✓ Positional Encoding
- ✓ SIREN-based Positional Encoding
- ✓ Results - SIREN-based Positional Encoding

What is coming?

- Local self-attention
- Differentiable span
- Results - Differentiable span

Self-attention is expensive

Intuition with numbers

- Image: 32x32
- Pixels (tokens): $N = 1024$
- Number of attention scores: $1024 * 1024 = 1,048,576$
- Self-attention costs: $O(N^2)$

True for other types of inputs (text, videos, etc)

Can we do better?

- Costs: lower
- Accuracy: same or higher

Local Self-attention - The idea

Assumptions:

- Tokens in a close neighborhood are more relevant
- Long-distance relations can be captured over multiple layers

Idea:

- Don't compute all attention scores (1024×1024)
- Only compute attention scores in a local neighborhood of a smaller size

Gain:

- Reduce computation costs $\rightarrow O(N)$

Accuracy?

- For many of the natural tasks, the assumptions hold, the accuracy is the same or higher

Local Self-Attention - Previous Work

Independent blocks of fixed size (hyperparameter)

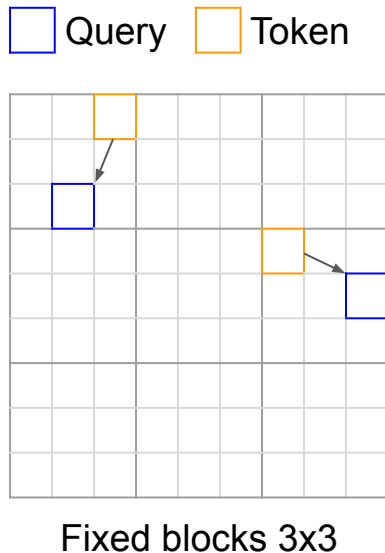
A query only attends to keys inside the same block *

Finding the right block size is difficult:

- Big enough to include relevant tokens
- Small enough to reduce computation costs

Can we do better?

Learn the size of attention span from data (Differentiable Span)

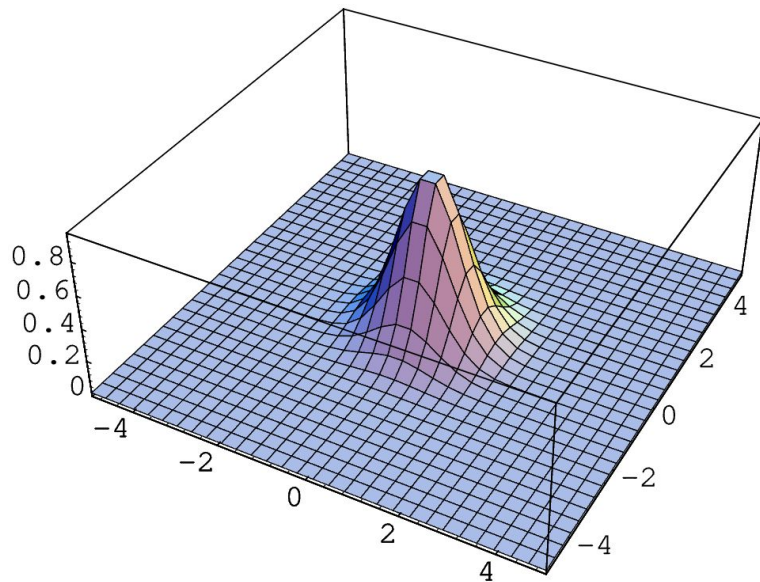


Differentiable Span - How?

We need a function that represents the locality feature:

- Close tokens are important = higher scores
- Farther tokens are less important = lower scores

Gaussian Function!

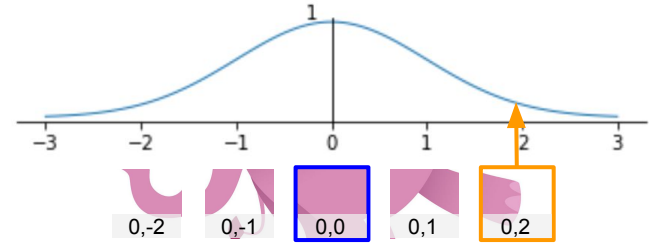


Differentiable Span - Gaussian Function

Gaussian function is **centered around each query**.

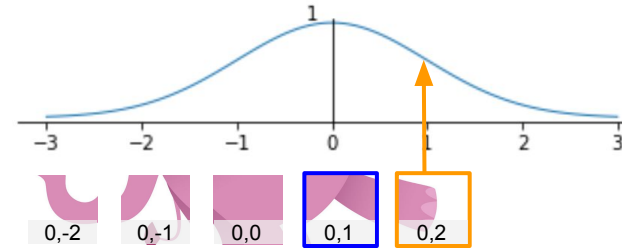
Modulate attention scores relevant to this query using the values of Gaussian Function

Modulate = Multiply



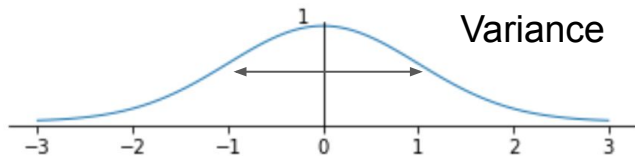
The attention scores for a query depend on:

- The content of the query and keys
- The locality feature (Gaussian Function)



Gaussian Function - Parameterization

Mean: 0 (always centered around a query)



General covariance matrix $C \in \mathbb{R}^{n \times n}$ must be positive semidefinite

- Difficult to parameterize
- Simplify by using diagonal covariance matrix

$$f(x) = e^{-x^T C x} \quad x \in \mathbb{R}^n$$

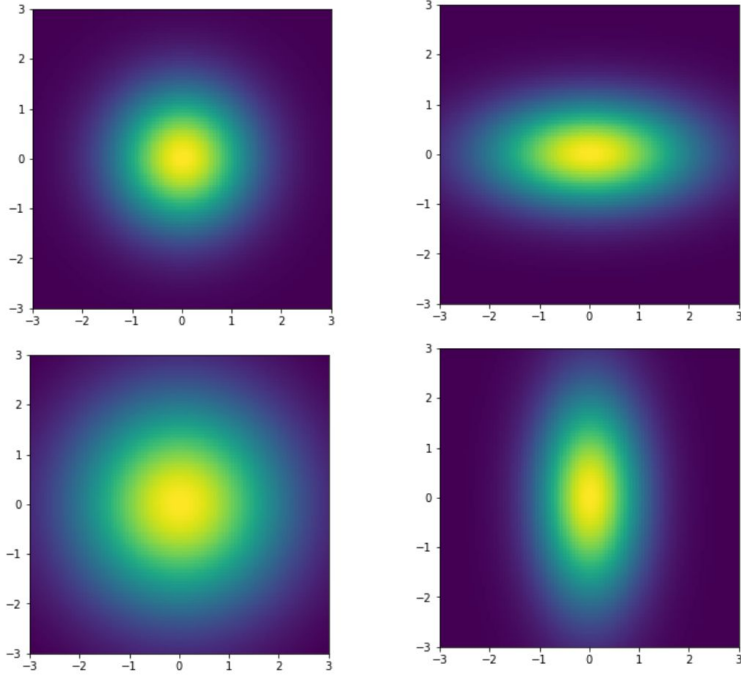
Multi-dimensional Gaussian Function

Diagonal covariance matrix:

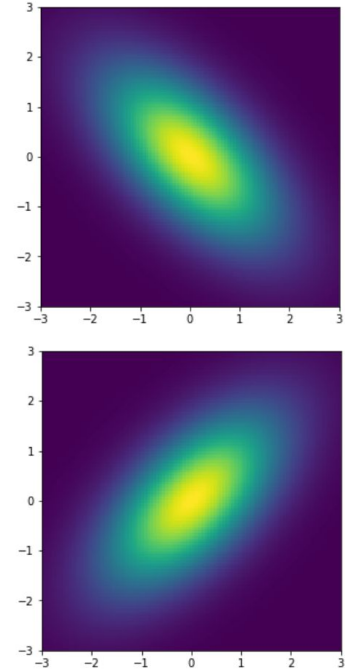
- Each dimension has an **independent standard deviation** parameter

Learnable parameters: $\sigma \in \mathbb{R}^n$

Diagonal Covariance Matrix



✓ Can represent these



✗ Can't represent these

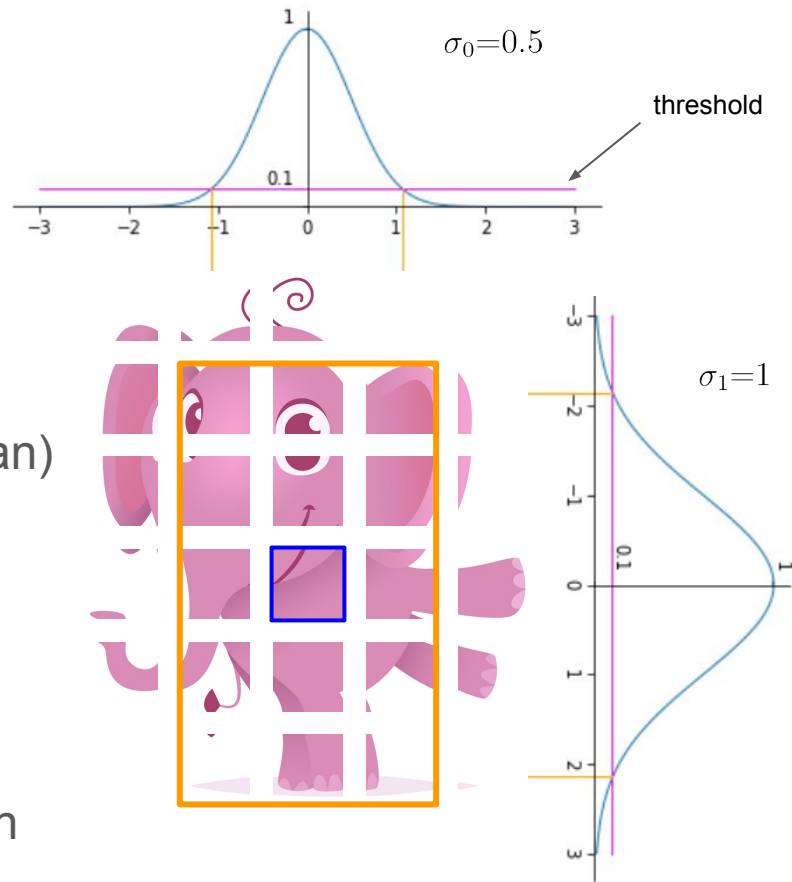
Gaussian Function + Threshold

Gaussian Function is **continuous**
= Gives values to all keys around a query

Define the relevant neighborhood (attention span)
using a **threshold**

Threshold is a constant: 0.1

Learned σ + constant threshold =
Learned attention span for Local Self-attention



How does Differentiable Span affect the accuracy?

Experiment 1 to compare: (Modulated attention scores)

- Regular scores: Positional Self-attention
- Modulated scores: Positional Self-attention + **Gaussian Function**

Validate the effectiveness of the **learned locality feature**

Experiment 2 to compare: (Modulated attention scores in limited neighborhood)

- All modulated scores: Positional Self-attention + Gaussian Function
- Local modulated scores: Positional Self-attention + Gaussian Function + **threshold**

Validate the possibility of **dropping scores outside** learned attention span

* 1D & 2D (SIREN-based)

Results - Differentiable Span without threshold

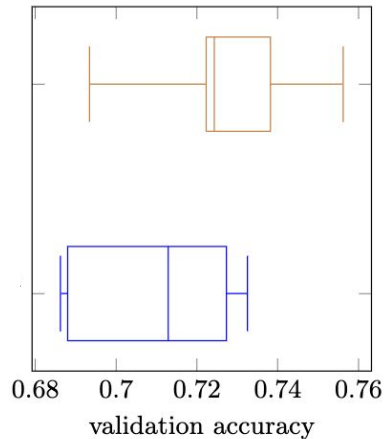
Regular scores vs Modulated scores

- 1D: lower accuracy
- 2D: higher accuracy

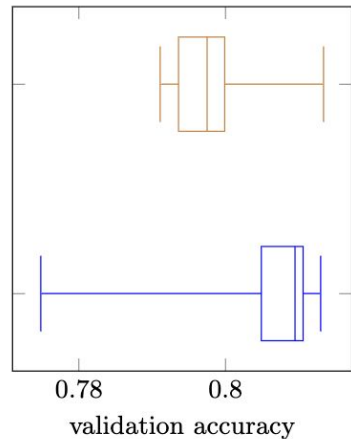
1D not enough to reflect the locality feature in the input (2D image).

Regular scores
(no differentiable span)

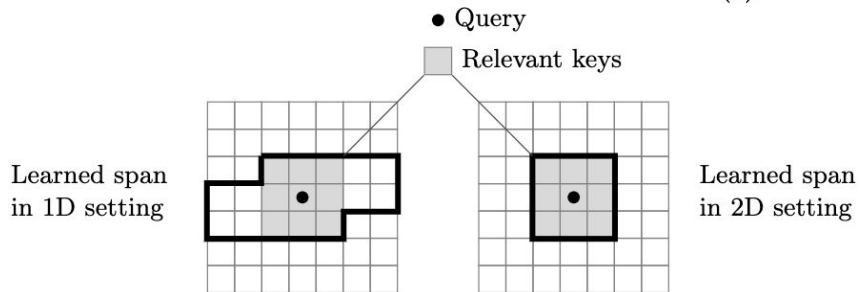
Modulated scores
(differentiable span without threshold)



(a) 1D



(b) 2D



Results - Differentiable span with threshold

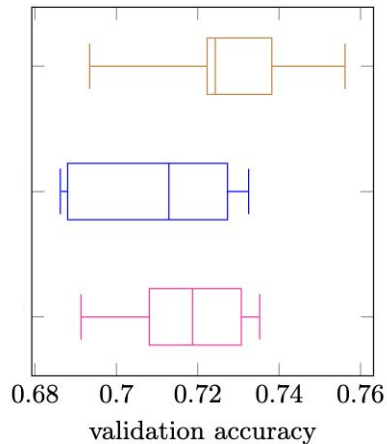
Applying the threshold is even better

- Removes noise from irrelevant tokens
- Parameter efficiency: Allow the network to focus on relations in the local area

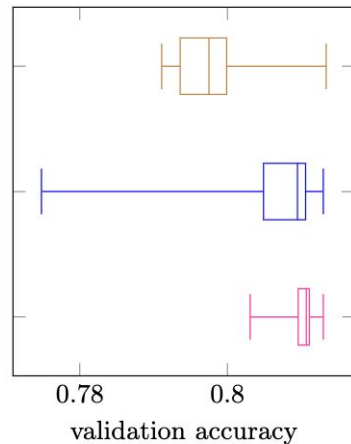
Regular scores
(no differentiable span)

Modulated scores
(differentiable span without threshold)

Modulated scores in learned span
(differentiable span with threshold)



(a) 1D



(b) 2D

Local Self-Attention + Differentiable Span

Learned span reduces the operations

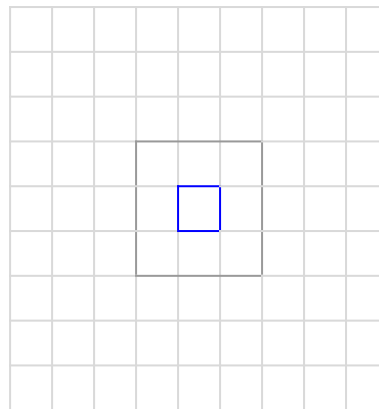
Doesn't guarantee $O(N)$

Different span per layer

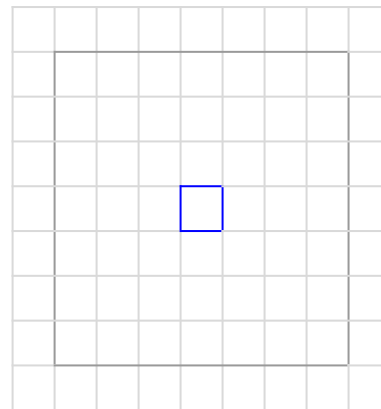
Image classification task

+ one token in the last layer for classification

- Early layers = smaller attention span
- Later layers = bigger attention span



Less operations



More operations

Recap Differentiable Span

We have shown that:

- The locality feature can be captured by Gaussian Function
- Modulated scores improves accuracy
(when modeled dimensions match input dimensions)
- Learned span **can be used** for Local Self-attention
(only compute attention scores inside the span)

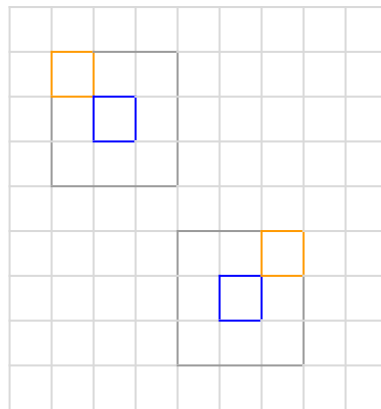
How to use learned span to save computations?

Ours is different that Previous Work

Previous work used **fixed** independent blocks

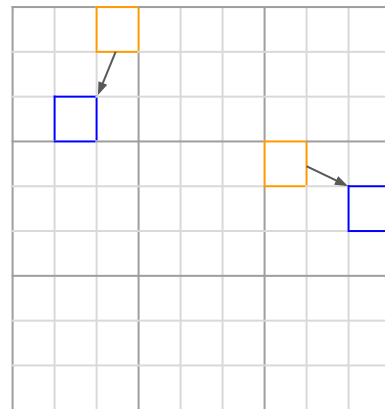
Ours use **relative** blocks centered around each query

How to efficiently limit the attention operation to this **moving span**?



Relative blocks 3x3
Ours

□ Query □ Token



Fixed blocks 3x3
Previous Work

We failed

OK we tried:

- Version 1: Compute all attention scores. Drop unneeded ones.
- Version 2: A tighter data layout. Compute as little as possible.
- Version 3: A tight layout, simpler but looser than Version 2.

V2 and V3 are designed to do less, but ended up taking more memory and is slower in practice.

A detailed analysis of why this is the case can be found in the paper.

There is still hope! (future work)

Our Contributions

SIREN-based Positional Encoding is promising

- Give comparable results but we **couldn't** show that they are better (future work)
- They are a promising choice for multi-dimensional positional encoding

Differentiable span can be used to

- Model the locality feature: **increases** the accuracy when a proper number of dimensions is used
- Learn attention span: **can** be used for local self-attention
- Provided two ways to use the learned span that **don't** reduce computation costs

Future work

SIREN-based positional encoding

- How to get SIREN-based to beat the baseline?
- Provide more stable parameterization (less sensitive to hyperparameters)

Differentiable Span

- Provide general parameterization of Gaussian Function
- Develop an efficient way to use the learned span in local self-attention

Thanks to

Supervisors:

- David W. Romero (weekly supervisor)
- Michael Cochez (first supervisor)
- Peter Bloem (second reader)

For the valuable support and feedback.

Marian (my partner) and friends who kept encouraging me!



Have questions?

Why a happy pink elephant?