Perceiver IO: A General Architecture for Structured Inputs & Outputs

陈冠华

PhD-y4 from The University of Hong Kong

Interests: Machine translation, unsupervised learning, cross-lingual text generation, multimodal,

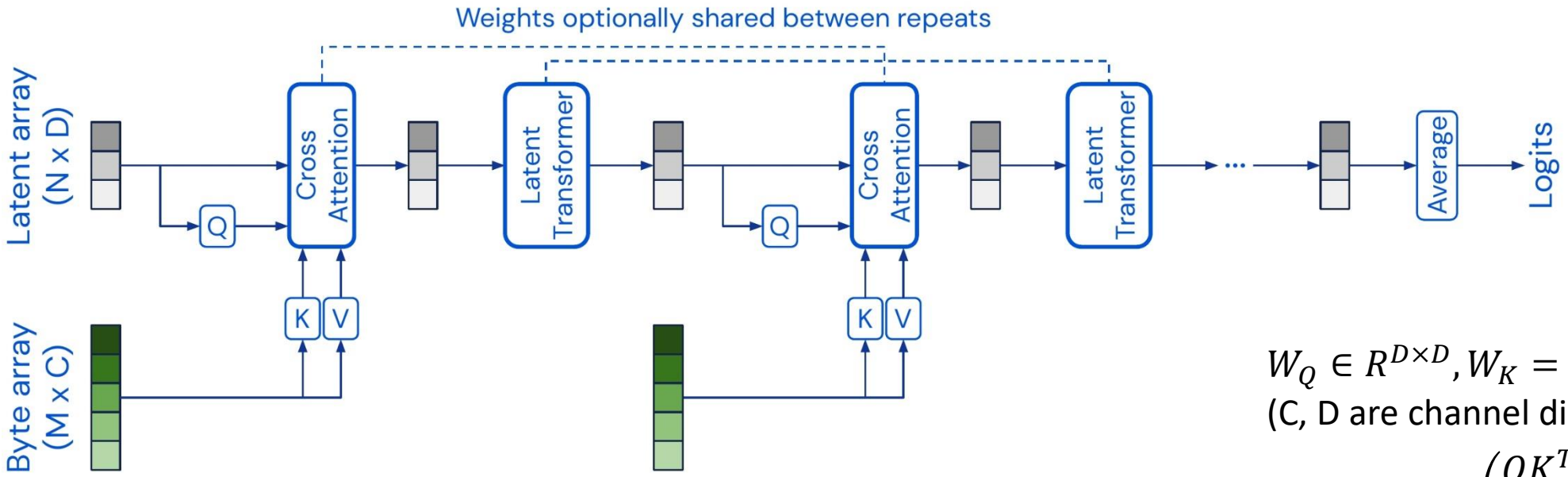
Background

- Perceiver: General Perception with Iterative Attention (ICML 2021)
  - Many modality-specific input (image, audio, video, point clouds, etc.)
  - Use an asymmetric attention mechanism to iteratively distill inputs into a tight latent bottleneck (a fixed-size latent space), allowing it to scale to handle very large inputs.
  - Decoupling the input size and the depth, and construct very large networks on large-scale data.

Background

Perceiver

- Iteratively attends to the input byte array by alternating cross-attention and latent self-attention blocks



$M \gg N$, N is fixed ($M = 50176$ for 224×224 ImageNet images, N=512)

$W_Q \in R^{D \times D}, W_K = W_V \in R^{C \times D}$
(C, D are channel dimensions)

$$y = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

$O(m^2) \rightarrow O(mn)$

Background

Perceiver

ResNet-50 (He et al., 2016)	77.6
ViT-B-16 (Dosovitskiy et al., 2021)	77.9
ResNet-50 (FF)	73.5
ViT-B-16 (FF)	76.7
Transformer (64x64, FF)	57.0
Perceiver (FF)	78.0

Table 1. Top-1 validation accuracy (in %) on ImageNet. Models that use 2D convolutions exploit domain-specific grid structure architecturally, while models that only use global attention do not.

However, Perceiver only works for classification task ☹

Perceiver IO

- Extend to more tasks with different outputs
- Can produce many outputs, each with arbitrary shape and structure

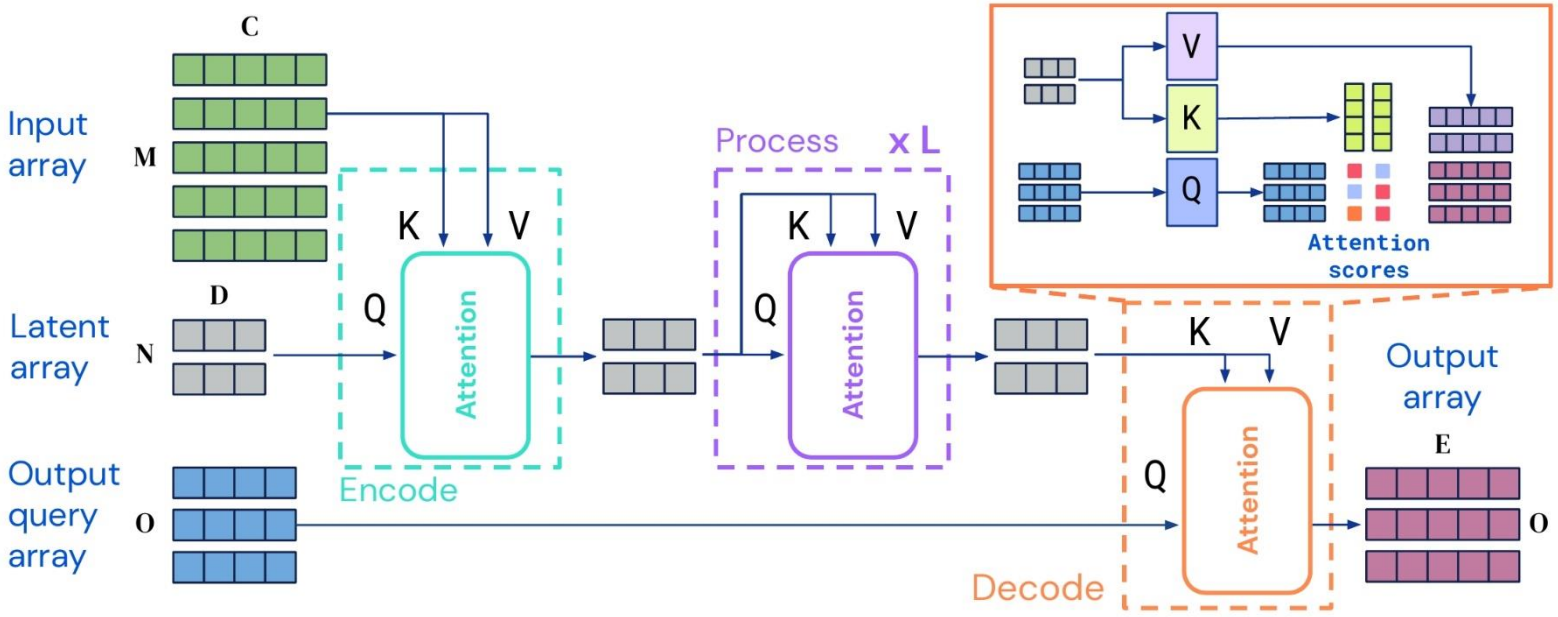


Figure 2: The Perceiver IO architecture. Perceiver IO maps arbitrary input arrays to arbitrary output arrays in a domain agnostic process. The bulk of the computation happens in a latent space whose size is typically smaller than the inputs and outputs, which makes the process computationally tractable even for very large inputs & outputs.

$$W_Q \in R^{E\times D}, W_K = W_V \in R^{D\times D}, W_O \in R^{D\times E}$$

$$y = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Perceiver IO

- Encode: using attention to map input arrays ($x \in R^{M \times C}$) to arrays in a latent space ($z \in R^{N \times D}$)
- Process: all arrays are in the same latent space ($z \in R^{N \times D}$)
- Decode: using attention to map latent arrays ($z \in R^{N \times D}$) to output arrays ($y \in R^{O \times E}$)

✓ No quadratic dependence on the input or output size
  ✓ Encoder and decoder attention modules depend linearly on the input and output size
  ✓ Latent attention is independent of both input and output sizes

Perceiver IO

Different output query array

- Reflect the downstream task and captures output structure
  - Spatial position in an image
  - Position of an output word in a sequence.

Modalities	Tasks	Preprocessing	Postprocessing	# Inputs	# Outputs
Text	Token-level pred.	Tokenization + Embed.	Linear projection	512×768	512×768
Text	Byte-level pred.	Embed.	None	$2,048 \times 768$	$2,048 \times 768$
Text	Multi-task (8 tasks)	Embed.	None	$2,048 \times 768$	8×768
Video	Flow prediction	Conv+maxpool	RAFT upsampling	$22,816 \times 64$	$11,408 \times 64$
Video+Audio+Label	Auto-encoding	Patch: 1x4x4 Vid, 16 Aud	Linear projections	$50,657 \times 704$	$803,297 \times 512$
StarCraft Unit Set	Encoding and Classification	Tokenization	Pointer network	512×256	512×128
Image	Classification	None	None	$50,176 \times 3$	$1 \times 1,000$

Table 1: We give details of each of the tasks we use to evaluate Perceiver IO here. The positional and task embeddings appended to inputs for each case are listed in Appendix Table 7.

Experiments

- Evaluated on a variety of tasks

Domain	Input Modality	Encoder KV input	Encoder KV channels	Decoder query input	Decoder query channels
Language (MLM)	Text	byte/token encoding + learned pos	768	learned pos	1280
Language (Perceiver IO++ MLM)	Text	byte/token encoding + learned pos	768	learned pos	1536
Language (GLUE)	Text	byte/token encoding + learned pos	768	Class query (per-task)	1280
Language (Perceiver IO++ GLUE)	Text	byte/token encoding + learned pos	768	Class query (per-task)	1536
Optical Flow	Video (image pairs)	[2 × conv features, 3D FFs]	450	[2 × conv features, 3D FFs]	450
Optical Flow (pixels)	Video (image pairs)	[Linear(2 × RGB), 2D FFs]	322	[Linear(2 × RGB), 2D FFs]	322
Kinetics	Video, Audio, Label	[Linear(RGB), 3D FFs, learned modality feat.] [sound pressure, 1D FF, learned modality feat.] [one-hot label, learned modality feat.]	704 704 704	[3D FFs, learned modality feat.] [1D FF, learned modality feat.] [learned modality feat.]	1026 1026 1026
StarCraft II	SC2 entities	Entity features	128	Entity features	128
ImageNet	Image	[RGB, 2D FFs]	261	Class query (single)	1024
ImageNet (learned pos)	Image	[Linear(RGB), learned pos]	512	Class query (single)	1024
ImageNet (conv)	Image	[Conv features, 2D FFs]	322	Class query (single)	1024

Table 7: The structure and size of the positional and task embeddings used to construct Perceiver IO's encoder key-value inputs and decoder query inputs, for each domain described in the main text. "[x, y]" indicates that x's and y's features are concatenated, while "x + y" indicates that x's and y's features are added to produce the full featurization. "FF" = Fourier features, as in [35].

Experiments

NLU tasks (GLUE benchmark)

- Enable deep models (26 layers and 40 layers)
- The dimension $C = E \neq D$
- Vocab: 32k (spm), 256+ bytes (UTF8 bytes)
- Using learnable position-dependent vectors to query the output of the final latent

Model	Perceiver IO Base	Perceiver IO	Perceiver IO++
Tokenizer	SentencePiece	UTF-8 bytes	UTF-8 bytes
Number of inputs (M)	512	2048	2048
Input embedding size (C)	768	768	768
Number of Process layers	26	26	40
Number of latents (N)	256	256	256
Latent size (D)	1280	1280	1536
FFW hidden dimension for latents	1280	1280	1536
Number of output queries during pretraining (O)	512	2048	2048
Dimension of learned queries (E)	768	768	768
FFW hidden dimension for outputs	768	768	768

Table 8: Perceiver IO architecture details for Language experiments

Experiments

NLU tasks (GLUE benchmark)

Model	Tokenization	N (# inputs)	M (# latents)	Depth	Params	FLOPs	**Average**
BERT Base (test) [21]	SentencePiece	512	512	12	110M	109B	81.0
BERT Base (ours)	SentencePiece	512	512	12	110M	109B	81.1
Perceiver IO Base	SentencePiece	512	256	26	223M	119B	**81.2**
BERT (matching FLOPs)	UTF-8 bytes	2048	2048	6	20M	130B	71.5
Perceiver IO	UTF-8 bytes	2048	256	26	201M	113B	81.0
Perceiver IO++	UTF-8 bytes	2048	256	40	425M	241B	**81.8**

Table 2: **Perceiver IO on language**: results on the GLUE benchmark (higher is better). Following [21] we exclude the WNLI task. We use Pearson correlation on STS-B, Matthews correlation on CoLa and accuracy on the remaining tasks.

Train a tokenizer-free language model that matches the performance of a baseline model trained with a SentencePiece tokenizer, hence removing the need for hand-crafted and potentially harmful tokenization schemes

Experiments

Multi-task Perceiver IO

- Single token shared among tasks (Shared input token)
- Task-specific tokens (Task specific input token)
- Learn a single query for each task (Multitask query)

Multitask method	**Avg.**
Single task	81.0
Shared input token	81.5
Task specific input token	**81.8**
Multitask Query	**81.8**

Table 3: Multitask Perceiver IO. Results use the same metric as Table 2 (higher is better). (GLUE benchmark)

Experiments

Image Classification

- Competitive with ViT family without relying on 2D convolutions
- Best result on ImageNet without 2D architectural or feature information (72.7)
- Better when adding a simple 2D conv+maxpool preprocessing network

Model	Pretrained?	Top-1 Acc.
ConvNet baselines		
ResNet-50 [30]	N	78.6
NFNet-F6+SAM [8]	N	86.5
Meta Pseudo Labels [60]	Y	90.2
ViT baselines		
ViT-B/16 [24]	N	77.9
ViT-H/14 (pretrained) [24]	Y	88.6
DeiT 1000 epochs [86]	N	85.2
CaiT-M48 448 [87]	N	86.5
w/ 2D Fourier features		
Perceiver	N	78.6
Perceiver IO	N	79.0
Perceiver IO (pretrained)	Y	84.5
w/ learned position features		
Perceiver (learned pos)	N	67.6
Perceiver IO (learned pos)	N	72.7
w/ 2D conv + maxpool preprocessing		
Perceiver (conv)	N	77.4
Perceiver IO (conv)	N	82.1

Table 6: Results on ImageNet image classification (top-1 accuracy, higher is better). The first two blocks show representative baseline results. "Perceiver" uses an average+project decoder while "Perceiver IO" uses a cross-attend decoder.

Conclusion

- Perceiver IO, a general-purpose neural network architecture
- Handling general purpose inputs and outputs while scaling linearly in both input and output sizes.
- Simplify the construction of sophisticated neural pipelines and facilitate progress on multimodal and multitask problems

Thank you