Continual Transformers: Redundancy-Free Attention for Online Inference

Lukas Hedegaard **Arian Bakhtiarnia** **Alexandros Iosifidis**
DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark
{lhm, arianbakh, ai}@ece.au.dk

Abstract

Transformers in their common form are inherently limited to operate on whole token sequences rather than on one token at a time. Consequently, their use during online inference on time-series data entails considerable redundancy due to the overlap in successive token sequences. In this work, we propose novel formulations of the Scaled Dot-Product Attention, which enable Transformers to perform efficient online token-by-token inference on a continual input stream. Importantly, our modifications are purely to the order of computations, while the outputs and learned weights are identical to those of the original Transformer Encoder. We validate our *Continual* Transformer Encoder with experiments on the THUMOS14, TVSeries and GTZAN datasets with remarkable results: Our *Continual* one- and two-block architectures reduce the floating point operations per prediction by up to 63× and 2.6×, respectively, while retaining predictive performance.

1 Introduction

Many real-life usage scenarios such as the perception in self-driving cars and live monitoring of critical resources process a continual stream of inputs and require near-instantaneous predictions per time-step. This stands in contrast to what many common benchmarks for deep learning evaluate, namely the operation on distinct batches of data with no inter-batch relationships. Consequently, a plethora of methods have been developed [1, 2, 3, 4, 5, 6, 7, 8], which focus on batch-wise processing, but fail to optimise for online operation, where new information (e.g., a video frame / token) arrives at each step from a continual input stream, and future information is not available at the current time-step. We need a class of networks, which operate efficiently on *both batches of data and on continual streams*.

Accordingly, we propose a reformulation of the Transformer Encoder as a Continual Inference Network (CIN, Section 2.1) which accelerates the stream processing on time-series data, while retaining weight-compatibility. Specifically, we derive two variants of Continual Scaled Dot-Product Attention (SDA) for the cases where prior output tokes *should* and *should not* be updated after observing a new input token. Notably, our attention formulations reduce the per-step cost of SDA [6] from time complexity $\mathcal{O}(n^2d)$ to $\mathcal{O}(nd)$ and memory complexity $\mathcal{O}(n^2)$ to $\mathcal{O}(nd)$ and are readily embedded into Continual Multi-Head Attention (MHA) and Continual Transformer Encoder blocks. Finally, we propose the use of Recycling Positional Encoding to accommodate progressive caching of partial attention results for continual data streams.

Due to the interdependence of SDA outputs, Continual Transformers are most efficient for shallow architectures. Shallow Transformers have many applications such as augmentations of CNNs [9], light-weight Natural Language Processing [10], fusion operations in multi-modal (e.g. audio-visual)

Preprint. Under review.

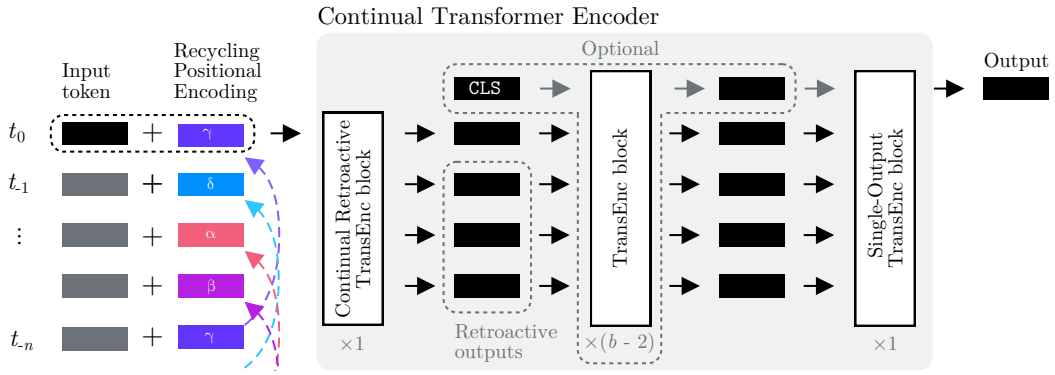


Figure 1: **Multi-block Continual Transformer Encoder with Recycling Positional Encoding.** For $b > 2$ blocks, regular Transformer Encoder blocks can be added between an initial Continual Retroactive block and a final Single-Output block. A class-token may be used after the initial block.

settings [11] and early exit branches in multi-exit architectures [12, 8]. In our experiments[1], we validate their exceptional efficiency improvements on common benchmarks in Online Action Detection [13] and Online Audio Classification [14].

2 Related Work

2.1 Continual Inference Networks

Definition (Continual Inference Network). *A Deep Neural Network, which*

- *is capable of continual step inference without computational redundancy,*
- *is capable of batch inference corresponding to a non-continual Neural Network,*
- *produces identical outputs for batch- and step inference given identical receptive fields,*
- *uses one set of trainable parameters for both batch and step inference.*

These requirements ensure that a Neural Network has broad applicability for both (offline) batch-wise inference (i.e., most research benchmarks) and online stream processing. While non-CINs can operate on streams of data by caching prior steps in a first-in first-out (FIFO) queue and aggregating them to a full (spatio-)temporal input, which is processed similarly to an offline batch, this entails computational redundancy in proportion with the sequence length. CINs perform step-wise inference without such caching and repeat computation. Uni-directional Recurrent Neural Networks are an example of Continual Inference Networks. Their default mode of operation is by time-step and they are easily applied to spatio-temporal batches of data by concatenation of the step-wise outputs. Recently, a modification to the spatio-temporal 3D convolution was proposed [15], which enables existing 3D CNNs to operate efficiently during continual inference. A similar principle was used to enhance Spatio-temporal Graph Convolutions as well [16]. In Section 3, we derive a CIN formulation for Transformer Encoders.

2.2 Transformer architectures

Initially proposed for sequence-to-sequence modelling in Natural Language Processing, the Transformer [6] has become a canonical building block in many applications of Deep Learning, including Computer Vision [17, 7, 18, 19] and Audio Classification [20]. Their success can be partly attributed to reduced inductive bias compared with CNNs and RNNs, which allows better adaptations when sufficiently large datasets are available; the Scaled Dot-Product Attention (SDA) maps a set of input tokens to a set of outputs without inherent preconceptions. However, this many-to-many attention exhibits quadratic growth in time and space complexity with the token count n in the set.

A great deal of research has sought to improve the efficiency of Transformers [21]. Block-wise or Chunking methods such as Image Transformer [22] and Vision Transformer [17] group up entities of a

[1]Source is code provided in supplementary material. Link will be made available upon acceptance.

local receptive field into a single block, reducing the $\mathcal{O}(n^2)$ complexity to $\mathcal{O}(n_b^2)$, where $n_b < n$ is the number of blocks. Techniques such as sliding windows, dilation and pooling can be used to achieve a similar effect [23]. The Reformer [24] reduces the complexity to $\mathcal{O}(n \log n)$ by learning groupings in a data-driven manner via Locality-Sensitive Hashing (LSH). A different paradigm aims to derive approximations of the self-attention matrix. Methods such as Linformer [25], Nyströmformer [26] and Performer [27] reduce the complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. Unlike these efforts, our approach produces the *exact* same computational outputs for temporal sequences as the original Multi-Head Attention.

3 Continual Transformers

The Scaled Dot-Product Attention (SDA) is central to the Transformer. Consider the case, where the query, key and value inputs to the SDA constitute a continual stream of d-dimensional tokens and we wish to compute the outputs for each step immediately considering $n-1$ prior tokens. Let us examine three SDA implementations and derive the complexity of each.

3.1 Regular Scaled Dot-Product Attention

Denoting query, key, and value sequence matrices by $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$, the regular Scaled Dot-Product Attention first defined by Vaswani et al. [6] can be written as:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V} \qquad \mathbf{A} = \exp\left(\mathbf{Q}\mathbf{K}^\top / \sqrt{d}\right) \qquad \mathbf{D} = \text{diag}\left(\mathbf{A}\mathbb{1}_n^\top\right), \tag{1}$$

where $\mathbf{A}, \mathbf{D} \in \mathbb{R}^{n \times n}$ and $\mathbb{1}_n$ is a row-vector of n ones. In each time-step, we can update $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ by discarding their oldest token and prepending a new one in a FIFO manner. This is a common implementation for step-wise inference, e.g. found in the FAIRSEQ library [28].

Each time-step results in $2n^2d + 2nd$ multiplications, $2n^2d - nd - n$ additions, and n^2 exponentiations as accounted for in Appendix A.1, which amounts to a time complexity of $\mathcal{O}(n^2d)$ and a $\mathcal{O}(n^2)$ memory complexity originating from the transient feature-map $\mathbf{A}$. Furthermore, a constant-sized cache of size $3(n-1)d$ is needed to store the $n-1$ latest tokens in $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$. We could avoid considerable redundancy by caching $\mathbf{Q}\mathbf{K}^\top$ directly. However, this comes with a memory penalty of $(n-1)^2$. Fortunately, another computational scheme can be devised.

3.2 Continual Retroactive Scaled Dot-Product Attention

We can compute $\mathbf{D}^{-1}\mathbf{A}\mathbf{V}$ in a step-wise manner using the latest query, key, and value steps, $\mathbf{q}_{\text{new}}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}} \in \mathbb{R}^{1 \times d}$, alongside appropriately cached partial results. The softmax normalisation with $\mathbf{D}^{-1}$ can be efficiently implemented via column-aligned element-wise multiplications (denoted by $\odot$ hereafter) of a column-vector $\mathbf{d} = \mathbf{A}\mathbb{1}_n^\top$. If we cache the $n-1$ values for the prior step tokens, i.e. $\mathbf{d}_{\text{mem}} = \mathbf{A}_{\text{prev}}^{(-n+1:-1)}\mathbb{1}_{n-1}^\top$, alongside $\mathbf{Q}$ and $\mathbf{K}$, we can define the step update as:

$$\mathbf{d}^{(-n+1:-1)} = \mathbf{d}_{\text{mem}}^{(-n+2:0)} - \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{old}}^\top\right) + \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{new}}^\top\right) \tag{2}$$

$$\mathbf{d}^{(0)} = \exp\left(\frac{\mathbf{q}_{\text{new}}}{\sqrt{d}}\left(\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}}\right)^\top\right)\mathbb{1}_n^\top, \tag{3}$$

where $\mathbf{Q}_{\text{mem}}$ ($\mathbf{K}_{\text{mem}}$) are the $n-1$ prior query (key) tokens, $\mathbf{k}_{\text{old}}$ is the key from n steps ago, and $\parallel$ denotes concatenation of matrices along the first dimension. Negative indices indicate prior time-steps. An update for $\mathbf{A}\mathbf{V}$ can likewise be defined as a function of the $n-1$ prior values $\mathbf{A}\mathbf{V}_{\text{mem}}$:

$$\mathbf{A}\mathbf{V}^{(-n+1:-1)} = \mathbf{A}\mathbf{V}_{\text{mem}}^{(-n+2:0)} - \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{old}}^\top\right)\mathbf{v}_{\text{old}} + \exp\left(\mathbf{Q}_{\text{mem}}\mathbf{k}_{\text{new}}^\top\right)\mathbf{v}_{\text{new}} \tag{4}$$

$$\mathbf{A}\mathbf{V}^{(0)} = \exp\left(\frac{\mathbf{q}_{\text{new}}}{\sqrt{d}}\left(\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}}\right)^\top\right)\left(\mathbf{V}_{\text{mem}} \parallel \mathbf{v}_{\text{new}}\right). \tag{5}$$

Finally, we compute the Continual Retroactive Attention output in the usual manner:

$$CoRe\text{Att}(\mathbf{q}_{\text{new}}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}}) = \mathbf{d}^{-1} \odot \mathbf{A}\mathbf{V}. \tag{6}$$

An visual depiction of these update steps is provided in Appendix A.2. A time-step can now be computed with $7nd + 2n - 3d$ multiplications, $6nd + 3n - 6d - 3$ additions, and $3n - 2$ exponentials. This time complexity of $\mathcal{O}(nd)$ per step and a $\mathcal{O}(nd)$ memory complexity is a significant improvement over the prior $\mathcal{O}(n^2d)$ and $\mathcal{O}(n^2)$ complexities in Section 3.1.

3.3 Continual Single-Output Scaled Dot-Product Attention

Both the Regular and Continual Retroactive Dot-Product Attentions produce attention outputs for the current step, as well as $n-1$ retroactively updated steps. In cases where retroactive updates are not needed, we can simplify the computation greatly via a Continual Single-Output Dot-Product Attention ($CoSi$Att). In essence, the regular SDA is reused, but prior values of $\mathbf{k}$ and $\mathbf{v}$ are cached between steps (as in [28]), and only the attention corresponding to a single query token $\mathbf{q}$ is computed:

$$CoSi\text{Att}(\mathbf{q}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}}) = \mathbf{a}\left(\mathbf{V}_{\text{mem}} \parallel \mathbf{v}_{\text{new}}\right) / \mathbf{a}\mathbb{1}_n^\top, \qquad \mathbf{a} = \exp\left(\frac{\mathbf{q}}{\sqrt{d}}\left(\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}}\right)^\top\right). \tag{7}$$

A step output is computed with $2nd + 2d$ multiplications, $2nd - d - 1$ additions, and n exponentials. The time- and memory complexities remain $\mathcal{O}(nd)$ per step. Using the (leading) query $\mathbf{q}_{\text{new}}$ as input, the attention is purely causal. Alternatively, prior (lagging) query vectors could be cached and used as query input, though this would introduce a network delay.

3.4 Comparison of Scaled Dot-Product Attentions

Assuming $n-1$ prior $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ steps have been calculated by the Continual SDA modules, and that $\mathbf{Q} = (\mathbf{Q}_{\text{mem}} \parallel \mathbf{q}_{\text{new}})$, $\mathbf{K} = (\mathbf{K}_{\text{mem}} \parallel \mathbf{k}_{\text{new}})$, and $\mathbf{V} = (\mathbf{V}_{\text{mem}} \parallel \mathbf{v}_{\text{new}})$, we have the correspondence:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})^{(t)} = CoRe\text{Att}(\mathbf{q}_{\text{new}}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}})^{(t)} = CoSi\text{Att}(\mathbf{q}_t, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}}) \tag{8}$$

Here, $\mathbf{q}_t$ is the t^{th} row of $\mathbf{Q}$, i.e. $\mathbf{Q}^{(t)}$. During stream processing, the complexity of the Continual Retroactive SDA scales significantly more favourably that the regular SDA. For example, the floating point operations (FLOPs) are reduced by $31\times$ when $n = d = 100$ and $308\times$ when $n = d = 1000$. If retroactive output updates are not needed, the Continual Single-Output SDA reduces FLOPs by respectively $100\times$ and $1000\times$. The scaling properties are detailed in Appendix A.1.

3.5 Continual Multi-Head Attention

Continual Scaled Dot-Product Attentions can replace regular SDA's directly in a Multi-Head Attention (MHA). Given a new query, key, and value, $\mathbf{q}, \mathbf{k}, \mathbf{v}$, the Continual MHA is defined as

$$Co\text{MHA}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \left(\Big\Vert_{i=0}^{h-1} Co\text{Att}(\mathbf{q}\mathbf{W}_Q^i, \mathbf{k}\mathbf{W}_K^i, \mathbf{v}\mathbf{W}_V^i)\right)\mathbf{W}_O, \tag{9}$$

where $\parallel$ denotes concatenation of h heads and $\mathbf{W}_Q^i, \mathbf{W}_K^i \in \mathbb{R}^{d \times d_K/h}$, $\mathbf{W}_V^i \in \mathbb{R}^{d \times d_V/h}$, and $\mathbf{W}_O \in \mathbb{R}^{d_V \times d_O}$ are projection matrices of head i. CoAtt can be either $CoRe$Att or $CoSi$Att.

3.6 Continual Transformer Encoder

A Continual MHA block can be integrated in a Continual Transformer Encoder block as follows:

$$\mathbf{z} = \text{LayerNorm}\left(\mathbf{y} + \text{FF}(\mathbf{y})\right), \qquad \mathbf{y} = \text{LayerNorm}\left(\text{Sel}(\mathbf{x}) + Co\text{MHA}(\mathbf{x}, \mathbf{x}, \mathbf{x})\right), \tag{10}$$

where $\mathbf{x}$ corresponds to the newest step input and $\text{Sel}(\cdot)$ selects a single (last) token of $\mathbf{x}$ if *CoSi*MHA is used, or selects all tokens otherwise. $\text{FF}(\cdot)$ is a two-layer feed-forward network with weights $\mathbf{W}_1, \mathbf{W}_2$, biases w_1, w_2, and a activation function $\sigma(\cdot)$, i.e. $\text{FF}(\mathbf{x}) = \sigma(\mathbf{x}\mathbf{W}_1 + w_1)\mathbf{W}_2 + w_2$. Aside from the residual selection, this is identical to common Transformer Encoder implementations [6, 17].

3.7 Recycling Positional Encoding

Since a Transformer Encoder does not provide positional bias, it is common to augment a token $\mathbf{x}_i$ with a positional encoding $\mathbf{p}$, i.e. $\tilde{\mathbf{x}}_i = \mathbf{x}_i \circ \mathbf{p}_i$, where $\circ$ could be addition or concatenation. In regular Transformers, the index i denotes a position in a sequence rather than a position in time. However, this static positional assignment is problematic in the context of continual inference; the last token at time $t = 0$ will be the next-to-last token at time $t = 1$, and thus in need of a different positional encoding than in the prior time-step. Instead, CINs require dynamic positions. There have been multiple prior

works [29, 30, 31] which create relative encodings by augmenting the SDA with positional offsets between query and keys. While such a modification to the continual attentions is possible, it hinders compatibility with the regular SDA. Instead, we use a *Recycling Positional Encoding* (RPE), which lets the positional encoding follow each token in time and recycles old encodings:

$$\tilde{\mathbf{x}}_t = \mathbf{x}_t + \mathbf{p}_{\tau_t}, \qquad \tau_t = (\tau_{t-1} + 1) \bmod T, \tag{11}$$

where T is the number of encodings. While RPE does not specify relative encodings explicitly, the absolute positional interpretation of each token changes dynamically when a new token arrives. In practice, the network learns relative, shift-invariant positional information by training with random τ_0 for each batch. Random shifts during training were recently explored in [32, 33, 34] as well. RPE can use either learned or predefined encodings. In the latter case, Cyclic Positional Encoding [35], a sinusoidal encoding inspired by Gray code, is a good fit. If we reuse the encoding immediately after an old token has "slided out", i.e. $T = n$, a token will have the same positional encoding relative to another whether it was m steps older or $n - m$ steps newer. The positional ambiguity can be avoided by extending the number of positional tokens to $T = 2n - 1$. We explore both options in Section 4.1.2.

3.8 Architectural considerations

Block count In Section 3.4, we observed an exact correspondence between the results of the continual and regular SDA layers. However, the correspondence does not necessarily hold for stacked layers. Consider the result of stacking two Continual Single-Output Transformer Encoder blocks. While the first block outputs a step t that is identical to that in a corresponding regular block, the second block would have been initialised with prior step-wise inputs, which were the result of prior input windows instead of the current one; the correspondence would not hold. Though it is not convertible to/from a regular Transformer Encoder, the stacked Single-Output Transformer Encoder architecture has the merit of efficiency. This was exploited in Transformer-XL [31]. Given a single step input, the Continual Retroactive Transformer Encoder block produces output tokens corresponding to the entire observed sequence inside the window. Due to this one-to-many input-output mapping, it is not possible to stack multiple such layers. Nevertheless, it can be used in conjunction with a Continual Single-Output Transformer Encoder with optional regular Transformer Encoder blocks in between as illustrated in Fig. 1. The Regular Transformer Encoder blocks in the resulting architecture have a significantly larger computational complexity than the Continual Retroactive and Single-Output blocks. Consequently, we recommend that Continual Transformer Encoders be used primarily in lightweight architectures with one or two blocks unless compatibility with non-continual Transformers is not required and only Single-output blocks are used.

Class token It is common to add a class token as input to transformers [36, 17], which accumulates information from other tokens prior to classification. However, it cannot be used naïvely with CINs, as this would effectively double the number of input steps. In practice, it can be employed in Continual multi-block Transformer Encoders as input to the second block (see Fig. 1), but this placement limits class token interaction with downstream layers. It can also be used for one-block Transformer Encoders if the value token is omitted as input.

Peak memory reduction trick The FLOPs for $\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ are exactly n times those of $CoSi\text{Att}(\mathbf{q}, \mathbf{k}_{\text{new}}, \mathbf{v}_{\text{new}})$. Comparing their memory complexity, the regular SDA is $\mathcal{O}(n^2)$, while the Single-output SDA is $\mathcal{O}(nd)$. In practical applications where system memory is limited, we may thus reduce the maximum memory requirement of the computational device at inference by up to d/n (assuming $n \gg d$) by computing each row of the attention individually. However, this may reduce throughput due to reduced parallelism.

4 Experiments

We provide case studies within two perception disciplines, namely Online Action Detection (Section 4.1) and Audio Classification (Section 4.2). In each case, we will start with a brief overview of the field, followed by experiments and results.

4.1 Online Action Detection

Online Action Detection (OAD) [37] entails the per-frame classification of human actions in a video stream as they happen without the ability to change prior predictions nor to use future information. This is fundamentally more restrictive than Temporal Action Localisation, where the whole video clip is processed before start and end frames of an action are determined [38, 39, 40, 41].

The dominant design in OAD works at the time of writing is to employ a two-stream Convolutional Neural Network as backbone for frame-wise feature extraction with RGB images as inputs in one stream and Optical Flow fields in the other [42, 43, 44, 18, 45][2]. On top of these, OAD methods encode temporal information and perform predictions per time-step, e.g. by means of RNNs [42, 43, 44] or Transformers [18, 45]. Alongside the action detection for the current frame, an action anticipation task may be learned in parallel by means of decoder structures, as this has been found to improve the primary OAD task.

Unlike RNNs, an output update for the regular SDA in a Transformer block cannot be naïvely computed for a single step by feeding successive video frames. Instead, prior step features must be cached, re-loaded and re-processed by the Transformer in each step in correspondence with a predefined window-size of prior steps. As laid out in Section 3.8, Continual Transformers are especially efficient when either one or two Continual Transformer Encoder blocks are used. Accordingly, we start our experiments with a set ablation studies to simplify a recent transformer-based architecture, the OadTR [18]. We further investigate the impact of ablating class token position and the use of Recycling Positional Encoding and compare different RPE schemes for Continual Transformers. Finally, we evaluate our configurations on two widely used OAD datasets, THUMOS14 [13] and TVSeries [37].

4.1.1 Experimental setup

The THUMOS14 dataset [13] for OAD has 200 and 213 validation and test videos, respectively, with frame-level class annotations across 20 classes. As in prior OAD works, the model is trained on the validation set and evaluated on the test set. Similar to [18] we use pre-extracted features from a two-stream Temporal Segment Network (TSN) [46] trained on ActivityNet v1.3 [47] or Kinetics-400 [2].

For TVSeries [37], the network learns on the train and validations sets (20 videos) and evaluates on the test set (7 videos) as in [18]. RGB and Optical Flow features were extraced using an MMAction2 [48] pipeline with ActivityNet v1.3 [47] and Kinetics-400 [2] pretrained TSN ResNet-50 [49] backbones. This is similar to the feature extraction process used by LSTR [45].

Following Wang et al. [18], we use a batch size of 128, sequence length 64, initial learning rate 10^{-4} with a factor ten reduction each epoch, alongside weight decay 10^{-4}, and dropout with probability 0.1. We report results using two epochs of training on a Nvidia RTX2080 Ti GPU. We track mean Average Precision (mAP) for THUMOS14 and calibrated mean Average Precision (cmAP) [37] for TVSeries, alongside FLOPs per prediction and parameters of the OAD module (feature extraction excluded). We report the mean $\pm$ standard deviation over five runs.

4.1.2 Ablation studies

Removing the Decoder As a first step to make an efficient Continual OadTR, we remove the decoder blocks used for action anticipation, which has a large impact on computational efficiency and the ease of transformation to a Continual Inference Network. The first two lines of Table 1a present the results of the removal. Contrary to the observations of Wang et al., we did not find any drop in accuracy when excluding the decoder. We do, however, gain a large reduction in FLOPs and model size; they were reduced to 58% and 30%, respectively. Given these computational improvements, we exclude the decoder in subsequent experiments.

(Re)moving the Class token Class tokens should not be input naively to the first Transformer Encoder layer of a CIN (see Section 3.8). Accordingly, we ablate its use and position. In cases where

[2]The feature extraction commonly used in Online Action Detection (OAD) works is in itself quite computationally costly. We consider the optimisation of the backbone as orthogonal future work and will follow the same feature extraction procedure as other OAD works at this time.

Table 1: **Ablation experiments** on THUMOS14 with TSN-Anet features. **Best** metrics are highlighted. '-' indicates that a particular feature was not used.

(a) **Class token** variations with OadTR. `CLS` pos. is the encoder block into which `CLS` is input.

Enc. blocks	Dec.	CLS pos.	mAP (%)	FLOPs (M)	Params (M)
3	✓	1	57.0$_{\pm 0.5}$	2445.6	74.7
3	-	1	57.0$_{\pm 0.4}$	1430.6	22.2
3	-	2	**57.3**$_{\pm \mathbf{0.7}}$	1423.5	22.2
3	-	3	56.7$_{\pm 0.6}$	1417.2	22.2
3	-	-	56.8$_{\pm 0.3}$	**1410.9**	22.2
2	-	1	56.5$_{\pm 0.5}$	1020.7	15.9
2	-	2	**56.7**$_{\pm \mathbf{0.3}}$	1014.5	15.9
2	-	-	56.6$_{\pm 0.3}$	**1008.1**	15.9
1	-	1	**57.1**$_{\pm \mathbf{0.6}}$	611.7	9.6
1	-	-	56.3$_{\pm 0.2}$	**605.5**	9.6

(b) **Positional encodings** variations for *Co*OadTr.

Enc. blocks	Re-cycling	Learn	Pos. tokens	mAP (%)	FLOPs (M)	Params (K)
2	-	✓	n	45.3$_{\pm 0.9}$	410.9	15832
2	✓	✓	n	56.4$_{\pm 0.3}$	410.9	15832
2	✓	✓	$2n-1$	56.0$_{\pm 0.5}$	410.9	15897
2	✓	-	n	55.8$_{\pm 1.0}$	410.9	**15767**
2	✓	-	$2n-1$	**56.8**$_{\pm \mathbf{0.4}}$	410.9	**15767**
1	-	✓	n	44.0$_{\pm 0.8}$	9.6	9535
1	✓	✓	n	55.6$_{\pm 0.3}$	9.6	9535
1	✓	✓	$2n-1$	55.6$_{\pm 0.3}$	9.6	9599
1	✓	-	n	54.4$_{\pm 1.8}$	9.6	**9469**
1	✓	-	$2n-1$	**56.1**$_{\pm \mathbf{0.7}}$	9.6	**9469**

it is removed, we predict on the token corresponding to the last input token. The results of varying `CLS` pos are noted in Table 1a. For the one-block architecture, the removal came with noticeable drop in mAP, while the two-block architecture saw small improvements when removing or introducing the class token later. For the three block model, the use of class tokens in block two achieved the highest mAP. Though it is commonly accepted, that class tokens should be introduced alongside other inputs in the first block, our results indicate that they can accumulate sufficient information with only one or two blocks, and that later stage introduction may work better in some applications. In general, the achieved mAP when varying `CLS` pos. and number of blocks are very similar to one another, while (re)moving the class token and reducing the block size both reduce computational complexity. This encourages the use of shallow Transformer Encoders over deeper ones as well as the removal of class tokens, as we do in the following experiments.

Positional Encodings We can transfer parameters from the simplified one- and two block OadTR to the corresponding Continual architecture, *Co*OadTR. Here, the one block version (*Co*OadTR-b1) uses *CoSi*MHA, and the two block model (*Co*OadTR-b2) uses *CoRe*MHA in the first block and Single-output MHA in the second. However, a regular positional encoding is not suited for continual inference (see Section 3.7). We evaluate the performance of using non-continual encodings for continual inference, as well as of our proposed Recycling Positional Encodings with fixed or learned parameters. In addition, we explore the impact of extending the number of tokens from n to $2n-1$ to avoid positional ambiguity. As seen in Table 1b), non-continual encoding used in the continual setting result in severe mAP drop. Recycling Positional Encodings alleviate this. Comparing learned and fixed encodings, we find the learned encodings to work better when the number of encoding tokens corresponds to the sequence length n and the fixed encoding to work best when positional ambiguity is alleviated by extending the number of tokens to $2n-1$. Fixed encoding with $2n-1$ tokens works best overall and is employed in subsequent experiments unless stated otherwise. There is no difference in FLOPs for either strategy, and the difference in parameter count is negligible.

4.1.3 Comparison with prior works

We evaluate the (*Co*)OadTR architectures on THUMOS14 and TVSeries with two sets of features as described in Section 4.1.1. Since no prior OAD works have reported complexity metrics, we measured the FLOPs for TRN [43] based on the publicly available source code to serve as a point of reference. The results of this benchmark are presented in Table 2 and Fig. 2. OadTR and our simplified (continual) one-block (b1) and two-block (b2) versions without decoder and class tokens generally achieve competitive precision in comparison with prior works, surpassing all but OadTR and LSTR. On THUMOS14, our reproduced OadTR results are slightly lower than originally reported [18][3], whereas achieved TVSeries results are higher[4]. The (*Co*)OadTR-b# architecture largely retain precision and allow significantly reduced FLOPs per prediction. Our proposed continual variants *Co*OadTR-b1 and *Co*OadTR-b2 reduce FLOPs by 255× and 6.1×, respectively, compared

[3]The reported 58.3% on THUMOS14 could not be reproduced using their publicly available code.

[4]We attribute our higher mcAP to differences in the feature extraction pipeline.

Table 2: **Online Action Detection** results. FLOPs per prediction are noted for inference on THUMOS14. The **best** and ***next-best*** metrics are highlighted.

Model	Feat.	THUMOS14 mAP (%)	TVSeries mcAP (%)	FLOPs (M)
RED [42]	A.Net	45.3	79.2	-
TRN [43]		47.2	83.7	1387.5
FATS [50]		51.6	81.7	-
IDN [44]		50.0	84.7	-
TFN [51]		55.7	85.0	-
LSTR [45]		**65.3**	88.1	-
OadTR [18]		***58.3***	85.4	2445.6
OadTR†		57.0±0.5	**88.6**±0.1	2445.6
OadTR-b2†		56.6±0.3	***88.3***±0.2	1008.1
OadTR-b1†		56.3±0.2	88.1±0.1	605.5
*Co*OadTR-b2 (ours)		56.8±0.4	87.7±0.6	***410.9***
*Co*OadTR-b1 (ours)		56.1±0.7	87.6±0.7	**9.6**
TRN [43]	Kin.	62.1	86.2	1462.0
FATS [50]		59.0	84.6	-
IDN [44]		60.3	86.1	-
PKD [52]		64.5	86.4	-
LSTR [45]		**69.5**	**89.1**	-
OadTR [18]		***65.2***	87.2	2513.5
OadTR†		64.2±0.3	***88.6***±0.1	2513.5
OadTR-b2†		64.5±0.5	88.3±0.2	1075.7
OadTR-b1†		63.9±0.5	88.1±0.1	673.0
*Co*OadTR-b2 (ours)		64.4±0.1	87.6±0.7	***411.9***
*Co*OadTR-b1 (ours)		64.2±0.4	87.7±0.4	**10.6**

†Using official source code or modifications there-off.

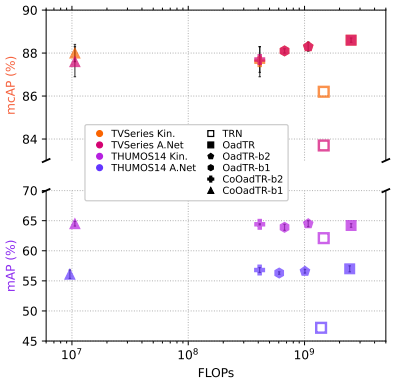


Figure 2: **Visual comparison** of OAD methods on THUMOS14 and TVSeries for backbones trained on ActivityNet 1.3 and Kinetics-400.

to OadTR. On average, continual and non-continual (*Co*)OadTR-b# models achieve similar mAP on THUMOS14, while OadTR-b# have slightly higher mcAP on TVSeries. We attribute these discrepancies to differences in positional encoding.

4.1.4 Audio-Visual Online Action Detection

To showcase the validity of our method in audio-visual settings as well, we explore the addition of audio-features to the Online Action Detection task on THUMOS14. As described in Section 4.2, audio-features are extracted using Mel spectrograms and an AudioSet pre-trained VGGish network [53] (output of the penultimate layer) on 1.0 second windows with a step size of 0.2 seconds to match the 5.0 FPS sampling rate of the video features.

The audio-features by themselves do not provide enough signal to reach good Online Action Detection performance (yielding only 6.7% mAP with an OadTR network). When concatenated with RGB and Flow they do provide a modest improvement as seen in Table 3. On average, this amounts to $+0.6\%$ mAP when combined with ActivityNet features and $+0.5\%$ mAP when used with Kinetics-400 features with shallower models enjoying the largest improvements.

4.2 Audio Classification

4.2.1 Background

Audio Classification is the categorisation of audio waveforms. Though waveform sequences can be used directly [54], it is common to first convert them to spectrograms. Mel spectrograms are obtained by a nonlinear transformation of a frequency scale [55], which is designed based on empirical knowledge about the human auditory system [56]. By employing spectrograms, audio classification can be approached in the same way as image classification [57].

4.2.2 Experiments

We conduct experiments on the Music Genre Classification dataset GTZAN [58]. It consists of 100 30-second clips for each of ten music genres. Each audio clip is sampled at 22,050 Hz. Since there

Table 3: **Audio-Visual** result, THUMOS14.

Model	Feat.	mAP (%)	FLOPs (M)
OadTR		**57.6**$_{\pm 0.6}$	2714.9
OadTR-b2	A.Net	***57.5***$_{\pm 0.5}$	1277.0
OadTR-b1	+	57.4$_{\pm 0.4}$	874.1
*Co*OadTR-b2	AudioSet	56.5$_{\pm 1.1}$	***415.0***
*Co*OadTR-b1		56.8$_{\pm 0.5}$	**13.8**
OadTR		64.4$_{\pm 0.4}$	2781.9
OadTR-b2	Kin.	**65.0**$_{\pm 0.4}$	1344.1
OadTR-b1	+	64.5$_{\pm 0.4}$	941.2
*Co*OadTR-b2	AudioSet	64.7$_{\pm 0.8}$	***416.0***
*Co*OadTR-b1		***64.8***$_{\pm 0.3}$	**14.8**

Table 4: **Audio Classification** results for GTZAN.

Method	Pos. Enc.	Acc. (%)	FLOPs (M)	Par. (K)
Maj. Voting	-	92.0	-	**0**
Trans-b2	learned	**95.0**$_{\pm 0.6}$	47.4	509
Trans-b1	learned	93.8$_{\pm 0.8}$	15.2	***286***
*Co*Trans-b2	fixed	***94.4***$_{\pm 1.0}$	27.0	509
*Co*Trans-b1	learned	93.2$_{\pm 1.1}$	***0.3***	***286***

are no predefined splits for GTZAN, we randomly select 10% of the data for validation and 10% for testing. The input is transformed to a temporal sequence by sliding a one-second window over each 30-second clip with a slide step size of 250ms, leading to 120 one-second clips. These are subsequently converted to Mel spectrograms. We then fine-tune a VGGish network, pre-trained on AudioSet [53] and use the penultimate layer for feature extraction. A batch size of 64 and the Adam optimizer [59] are used with an initial learning rate of 10^{-4}. The learning rate is reduced by a factor of 0.6 on plateau with a tolerance of two epochs, and an early stopping mechanism, where a maximum of 100 epochs are allowed. The VGGIsh base-network attains an accuracy of 86.1% on the dataset of one-second clips with 72.1M parameters and 864.7M FLOPs. Subsequently, the audio features are passed to a (Continual) Transformer Encoder which has 16 attention heads, an embedding dimension of 192 and an MLP dimension of 384. The Transformer Encoder is trained on the whole temporal sequence using a batch size of 32 and the AdamW optimizer [60] with a learning rate of 10^{-5} and a weight decay of 10^{-4} for 50 epochs. Since the Transformer Encoder is trained on entire 30-second clips, there are less data points available for this training. Accordingly, the size of the validation set is increased to 18%. All audio classification training procedures were carried out on a single Nvidia RTX 2080 Ti GPU. Table 4 presents the accuracy and efficiency of regular and Continual Transformers during online inference. As a baseline, we also include the result of majority voting among the clips to classify the entire sequence. The Continual Transformers obtain similar accuracy as regular a Transformers while consuming $1.76\times$ less FLOPs when using two blocks and $51.5\times$ less FLOPs when using one Transformer Encoder block.

5 Conclusion

In this work, we presented Continual Transformers, a redundancy-free reformulation of Transformers tailored for online inference. Central to the Continual Transformer are the Continual Retroactive and Single-Output Attention operations, which produce outputs identical to the original Scaled Dot-Product Attention for continual input sequences, while greatly reducing the time and memory complexity per prediction. The applicability of Continual Transformer architectures was experimentally validated in Online Action Detection and Online Audio Classification settings, observing upwards of multiple orders of magnitude reduction in time complexity for lightweight architectures at modest accuracy concessions. Continual Transformers constitute an algorithmic innovation, which could make possible hitherto unseen precision, speed, and power efficiency in online inference use-cases. With applications spanning enhanced perception and reactivity of robots and autonomous vehicles, weather forecasting, price prediction and surveillance, we hope it will be used for the common good.

Acknowledgments and Disclosure of Funding

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR).

References

[1] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. "3D Convolutional Neural Networks for Human Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.1 (2013), pp. 221–231. DOI: 10.1109/TPAMI.2012.59.

[2] J. Carreira and A. Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4724–4733.

[3] Gül Varol, Ivan Laptev, and Cordelia Schmid. "Long-Term Temporal Convolutions for Action Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.6 (2018), pp. 1510–1517. DOI: 10.1109/TPAMI.2017.2712608.

[4] Sijie Yan, Yuanjun Xiong, and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *AAAI Conference on Artificial Intelligence*. 2018, pp. 7444–7452.

[5] Negar Heidari and Alexandras Iosifidis. "Progressive Spatio-Temporal Graph Convolutional Network for Skeleton-Based Human Action Recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 3220–3224.

[6] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017, pp. 5998–6008.

[7] Anurag Arnab et al. "ViViT: A Video Vision Transformer". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[8] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. "Multi-Exit Vision Transformer for Dynamic Inference". In: *British Machine Vision Conference (BMVC)* (2021).

[9] Hugo Touvron et al. "Augmenting Convolutional networks with attention-based aggregation". In: *preprint, arXiv:2112.13692* abs/2112.13692 (2021).

[10] Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. "SMArT: Training Shallow Memory-aware Transformers for Robotic Explainability". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1128–1134. DOI: 10.1109/ICRA40945.2020.9196653.

[11] Kateryna Chumachenko, Alexandros Iosifidis, and Moncef Gabbouj. "Self-attention fusion for audiovisual emotion recognition with incomplete data". In: *arXiv preprint arXiv:2201.11095* (2022).

[12] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. "Single-Layer Vision Transformers for More Accurate Early Exits with Less Overhead". In: *preprint, arXiv:2105.09121* (2021).

[13] Haroon Idrees et al. "The THUMOS challenge on action recognition for videos "in the wild"". In: *Computer Vision and Image Understanding* 155 (2017), pp. 1–23. ISSN: 1077-3142.

[14] George Tzanetakis, Georg Essl, and Perry Cook. *Automatic Musical Genre Classification Of Audio Signals*. 2001. URL: http://ismir2001.ismir.net/pdf/tzanetakis.pdf.

[15] Lukas Hedegaard and Alexandros Iosifidis. "Continual 3D Convolutional Neural Networks for Real-time Processing of Videos". In: *preprint, arXiv:2106.00050* (2021). Apache 2.0 Licence.

[16] Lukas Hedegaard, Negar Heidari, and Alexandros Iosifidis. "Online Skeleton-based Action Recognition with Continual Spatio-Temporal Graph Convolutional Networks". In: *preprint, arXiv: 2203.11009* (2022). Apache 2.0 Licence.

[17] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations (ICLR)*. 2021.

[18] Xiang Wang et al. "OadTR: Online Action Detection with Transformers". In: *International Conference on Computer Vision (ICCV)* (2021). MIT License. URL: https://github.com/wangxiang1230/OadTR.

[19] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *European Conference on Computer Vision (ECCV)*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. 2020, pp. 213–229.

[20] Yuan Gong, Yu-An Chung, and James Glass. "AST: Audio Spectrogram Transformer". In: *Proc. Interspeech 2021*. 2021, pp. 571–575. DOI: 10.21437/Interspeech.2021-698.

[21] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. "Efficient Transformers: A Survey". In: *arXiv:2009.06732* (2020).

[22] Niki Parmar et al. “Image Transformer”. In: *International Conference on Machine Learning (ICML)*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 4055–4064.

[23] Iz Beltagy, Matthew E. Peters, and Arman Cohan. “Longformer: The Long-Document Transformer”. In: *arXiv:2004.05150* (2020).

[24] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. “Reformer: The Efficient Transformer”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[25] Sinong Wang et al. “Linformer: Self-Attention with Linear Complexity”. In: *arXiv:2006.04768* (2020).

[26] Yunyang Xiong et al. “Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2021).

[27] Krzysztof Marcin Choromanski et al. “Rethinking Attention with Performers”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[28] Myle Ott et al. “fairseq: A Fast, Extensible Toolkit for Sequence Modeling”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 48–53. DOI: 10.18653/v1/N19-4009.

[29] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. “Self-Attention with Relative Position Representations”. In: *North American Chapter of the Association for Computational Linguistics (NAACL)*. 2018.

[30] Cheng-Zhi Anna Huang et al. “Music Transformer: Generating Music with Long-Term Structure”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[31] Zihang Dai et al. “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, July 2019, pp. 2978–2988. DOI: 10.18653/v1/P19-1285.

[32] Shun Kiyono, Sosuke Kobayashi, Jun Suzuki, and Kentaro Inui. “SHAPE: Shifted Absolute Position Embedding for Transformers”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3309–3321. DOI: 10.18653/v1/2021.emnlp-main.266.

[33] Tatiana Likhomanenko et al. “CAPE: Encoding Relative Positions with Continuous Augmented Positional Embeddings”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 16079–16092.

[34] Mostafa Dehghani et al. “Universal Transformers”. In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyzdRiR9Y7.

[35] Yining Ma et al. “Learning to Iteratively Solve Routing Problems with Dual-Aspect Collaborative Transformer”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021.

[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[37] Roeland De Geest et al. “Online Action Detection”. In: *European Conference on Computer Vision (ECCV)*. 2016, pp. 269–284.

[38] Zheng Shou, Dongang Wang, and Shih-Fu Chang. “Temporal Action Localization in Untrimmed Videos via Multi-stage CNNs”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1049–1058. DOI: 10.1109/CVPR.2016.119.

[39] Huijuan Xu, Abir Das, and Kate Saenko. “R-C3D: Region Convolutional 3D Network for Temporal Activity Detection”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5794–5803. DOI: 10.1109/ICCV.2017.617.

[40] Zheng Shou et al. “CDC: Convolutional-De-Convolutional Networks for Precise Temporal Action Localization in Untrimmed Videos”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1417–1426. DOI: 10.1109/CVPR.2017.155.

[41] Chao-Yuan Wu et al. "Long-Term Feature Banks for Detailed Video Understanding". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 284–293. DOI: 10.1109/CVPR.2019.00037.

[42] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. "RED: Reinforced Encoder-Decoder Networks for Action Anticipation". In: *British Machine Vision Conference (BMVC)*. 2017.

[43] Mingze Xu et al. "Temporal Recurrent Networks for Online Action Detection". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. MIT Licence. 2019, pp. 5531–5540. DOI: 10.1109/ICCV.2019.00563. URL: https://github.com/xumingze0308/TRN.pytorch.

[44] H. Eun et al. "Learning to Discriminate Information for Online Action Detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 806–815.

[45] Mingze Xu et al. "Long Short-Term Transformer for Online Action Detection". In: *Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

[46] Limin Wang et al. "Temporal Segment Networks for Action Recognition in Videos". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.11 (2019), pp. 2740–2755. DOI: 10.1109/TPAMI.2018.2868668.

[47] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. "ActivityNet: A large-scale video benchmark for human activity understanding". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 961–970. DOI: 10.1109/CVPR.2015.7298698.

[48] MMAction2 Contributors. *OpenMMLab's Next Generation Video Understanding Toolbox and Benchmark*. https://github.com/open-mmlab/mmaction2. Apache 2.0 License. 2020.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90.

[50] Young Hwi Kim, Seonghyeon Nam, and Seon Joo Kim. "Temporally smooth online action detection using cycle-consistent future anticipation". In: *Pattern Recognition* 116 (2021), p. 107954. ISSN: 0031-3203.

[51] Hyunjun Eun et al. "Temporal filtering networks for online action detection". In: *Pattern Recognition* 111 (2021), p. 107695.

[52] Peisen Zhao et al. "Privileged Knowledge Distillation for Online Action Detection". In: *preprint, arXiv:2011.09158* abs/2011.09158 (2020).

[53] Shawn Hershey et al. "CNN architectures for large-scale audio classification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2017. DOI: 10.1109/icassp.2017.7952132.

[54] Jongpil Lee, Taejun Kim, Jiyoung Park, and Juhan Nam. "Raw Waveform-based Audio Classification Using Sample-level CNN Architectures". In: *NIPS, Machine Learning for Audio Signal Processing Workshop (ML4Audio)* (2017).

[55] S. S. Stevens, J. Volkmann, and E. B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *The Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. DOI: 10.1121/1.1915893.

[56] Keunwoo Choi, George Fazekas, and Mark Sandler. "Automatic tagging using deep convolutional neural networks". In: *International Society of Music Information Retrieval Conference (ISMIR)* (2016).

[57] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. "Rethinking CNN Models for Audio Classification". In: *arXiv:2007.11154* (2020).

[58] G. Tzanetakis and P. Cook. "Musical genre classification of audio signals". In: *IEEE Transactions on Speech and Audio Processing* 10.5 (July 2002), pp. 293–302. DOI: 10.1109/tsa.2002.800560.

[59] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[60] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations (ICLR)*. 2019.

A Appendix

A.1 Scaling properties of Continual and regular Multi-head Attention

A detailed account for the floating point operations involved in computing Regular-, Continual Retroactive-, and Single-output Scaled Product Attentions is given in Tables 5, 6, and 7.

Table 5: **Floating Point Operations** for the Scaled Dot-Product Attention in Eq. (1). $\mathbf{D}^{-1}(\cdot)$ can be efficiently computed as element-wise multiplication with $\mathbf{AV}$.

	Mul.	Add	Exp
Eq. (1.1)	$n^2d + nd$	$nd(n-1)$	0
Eq. (1.2)	$n^2d + nd$	$n^2(d-1)$	n^2
Eq. (1.3)	0	$n(n-1)$	0

Table 6: **Floating Point Operations** for the Continual Retroactive Dot-Product Attention in Eqs. (2) to (6). The outputs of the exponentials in Eq. (2) and Eq. (3) can be reused in Eq. (4) and Eq. (5) respectively, and are omitted in the count.

	Mul.	Add	Exp
Eq. (2)	$2(n-1)d$	$2(n-2)d + 2(n-1)$	$2(n-1)$
Eq. (3)	$nd + n + d$	$nd + (n-1) + d$	n
Eq. (4)	$2(n-1)d$	$2(n-1)d$	0
Eq. (5)	nd	$(n-1)d$	0
Eq. (6)	$nd + n$	0	0

Table 7: **Floating Point Operations** for the Continual Single-Output SDA in Eq. (7).

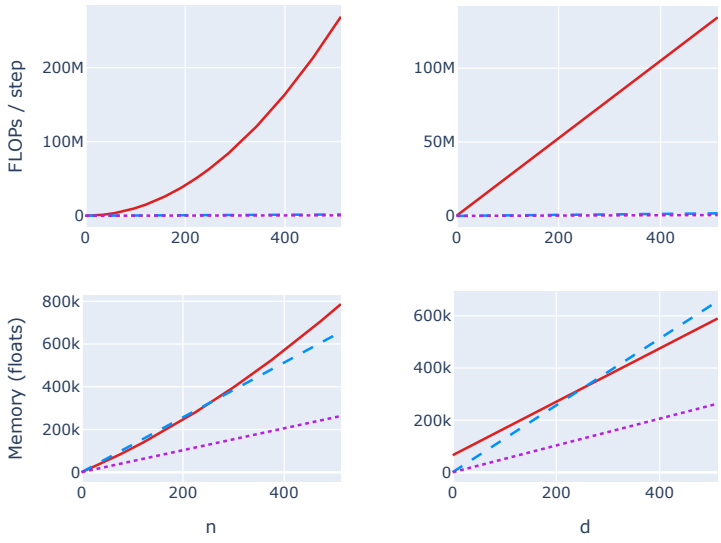
	Mul.	Add	Exp
Eq. (7.1)	$nd + d$	$(n-1)d + n - 1$	0
Eq. (7.2)	$nd + d$	$n(d-1)$	n

Fig. 3 illustrates the scaling of FLOPs and memory footprint with increasing sequence length n and embedding dimension d. Here, the Continual Retroactive and Single-Output SDAs spend significantly less FLOPs than the Regular SDA, which scales $\mathcal{O}(n^2)$ as opposed to $\mathcal{O}(nd)$ the continual variants. The Continual Single-Output SDA reduces memory footprint for all value combinations, and the Continual Retroactive SDA does so when $n \gtrapprox d$.

A.2 Supplemental visualisations

For the visually inclined, we supply a complementary graphical depictions of the Continual Retroactive SDA corresponding to Eqs. (2) to (6) in Fig. 4 and the Single-Output SDA in Eq. (7) in Fig. 5.

A schematic illustration of the Audio Classification experiments architecture is depicted in Fig. 6.



(a) $d = 256$ (b) $n = 256$

Figure 3: **FLOPs/step and memory footprint** for Regular, Continual Retroactive, and Continual Single-Output Scaled Dot-Product Attention at varying sequence length n and embedding dimension d. Column (a) has d fixed to 256; Column (b) has n fixed to 256.

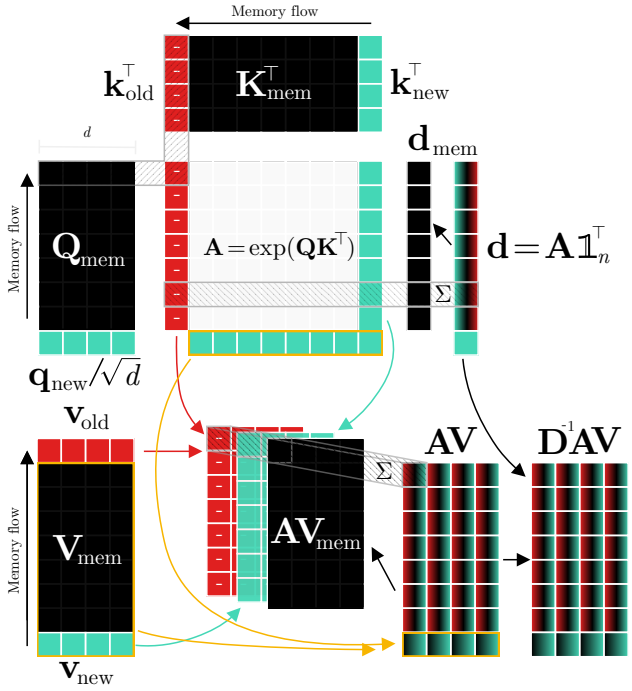


Figure 4: **Continual Retroactive Dot-Product Attention**. The query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) matrices are aggregated over time by caching the step vectors $\mathbf{q}_{\text{new}}$, $\mathbf{k}_{\text{new}}$, and $\mathbf{v}_{\text{new}}$ in each their FIFO queue (denoted by $\square_{\text{mem}}$). During each step, only the entries of $\mathbf{A}$ associated with $\mathbf{q}_{\text{new}}$, $\mathbf{k}_{\text{new}}$ and the oldest $\mathbf{K}$ step, $\mathbf{k}_{\text{old}}$ are computed. The diagonal entries of the row-normalisation matrix $\mathbf{D}$ as well as the $\mathbf{AV}$ can be updated retroactively by subtracting features corresponding to $\mathbf{k}_{\text{old}}$ and adding features related to $\mathbf{k}_{\text{new}}$ to the cached outputs of the previous step, $\mathbf{D}_{\text{mem}}$ and $\mathbf{AV}_{\text{mem}}$, respectively.

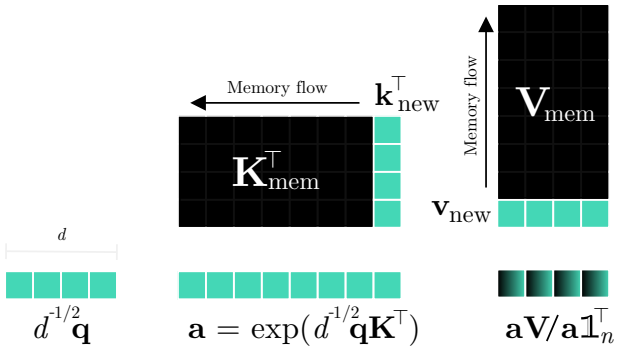


Figure 5: **Continual Single-Output Dot-Product Attention**. The key ($\mathbf{K}$) and value ($\mathbf{V}$) matrices are aggregated over time by caching the step vectors $\mathbf{k}_{\text{new}}$ and $\mathbf{v}_{\text{new}}$ in a FIFO queue. During each step, only the attention output associated with $\mathbf{q}$ is computed.

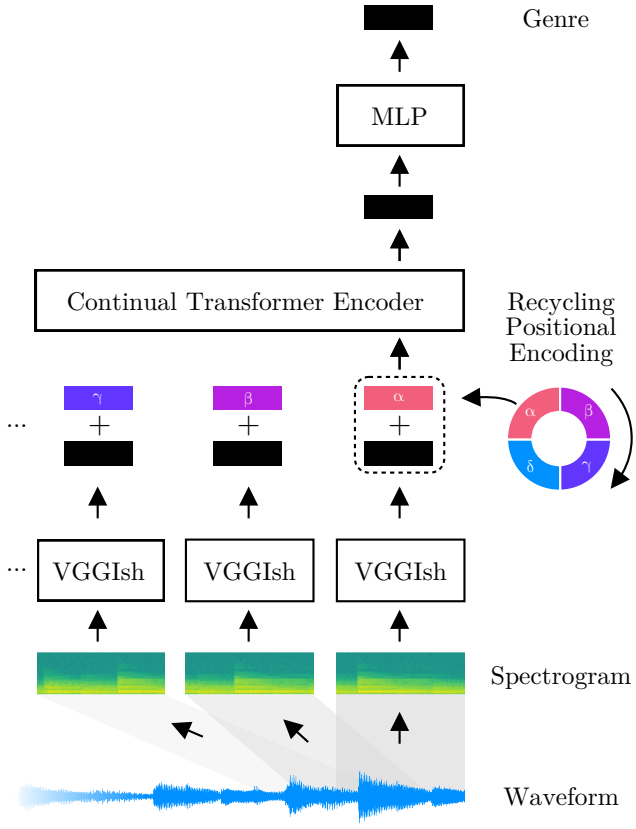


Figure 6: **Audio Classification Architecture**.