SOI: Scaling Down Computational Complexity by Estimating Partial States of the Model

Grzegorz Stefański[1], Paweł Daniluk[2], Artur Szumaczuk[2], Jakub Tkaczuk[2]
[1]Samsung AI Center Warsaw   [2]Samsung R&D Institute Poland
{g.stefanski, p.daniluk, a.szumaczuk, j.tkaczuk}@samsung.com

Abstract

Consumer electronics used to follow the miniaturization trend described by Moore's Law. Despite increased processing power in Microcontroller Units (MCUs), MCUs used in the smallest appliances are still not capable of running even moderately big, state-of-the-art artificial neural networks (ANNs) especially in time-sensitive scenarios. In this work, we present a novel method called Scattered Online Inference (SOI) that aims to reduce the computational complexity of ANNs. SOI leverages the continuity and seasonality of time-series data and model predictions, enabling extrapolation for processing speed improvements, particularly in deeper layers. By applying compression, SOI generates more general inner partial states of ANN, allowing skipping full model recalculation at each inference.

1   Introduction

Moore's Law (Moore, 1998) predicts that the number of transistors on a microchip will double every 2 years. As the number of transistors increases, the computational capacity of systems-on-chip (SoCs) also grows. Accompanied by the further miniaturization of components, the increased complexity of SoCs allows for the production of more compact appliances with equal or higher capabilities.

Despite the continuous development of more advanced hardware, the smallest appliances still cannot fully benefit from increasingly popular neural-based solutions, as they are not able to run them efficiently. Examples of such appliances include wireless in-ear headphones, smart watches, and AR glasses. Furthermore, the size of state-of-the-art neural networks is growing at much faster rate than that described by Moore's Law (Xu et al., 2018). Adding to the challenge, according to researchers (Leiserson et al., 2020), Moore's Law is starting to decelerate due to the physical constraints of semiconductors, and the "room at the bottom" is depleting.

Currently available neural network technologies enable machines to outperform humans in numerous applications in terms of measured performance (Nguyen et al., 2020; Bakhtin et al., 2022; Schrittwieser et al., 2020). However, despite this achievement, the same models fall significantly short when it comes to energy efficiency compared to humans. The human brain consumes a mere 20 Watts of power (Laughlin et al., 1998; Sengupta & Stemmler, 2014; Balasubramanian, 2021) and according to Xu et al. (2018) it is estimated to be over five orders of magnitude more energy efficient than modern neural networks.

This disparity in energy efficiency can be attributed to the common pursuit of the highest model quality in the literature, showcasing the full capabilities of the developed technology, often at the expense of efficiency. This behavior is justifiable due to the ease of comparing different solutions using well-defined metrics that are independent of the hardware and software. Conversely, estimating a model's energy efficiency is a more complex task, influenced by various factors including the running environment.

However, this trend within our community may prove restrictive, particularly for applications like real-time systems. These applications naturally demand optimal performance alongside high-efficiency solutions, rendering most current Deep Neural Networks (DNNs) impractical for such tasks. Furthermore, due to the substantial discrepancy between assumptions for high-efficiency on-device solutions and high-performing monolithic models, compressing these models may not be a viable means of applying state-of-the-art DNNs to real-time tasks.

The importance of neural system efficiency is also increasingly significant from ecological and economic standpoints (Lacoste et al., 2019; Patterson et al., 2021).

1.1 Related Works

The Short-Term Memory Convolution (STMC) (Stefański et al., 2023) was devised to enhance the efficiency of Convolutional Neural Networks (CNNs) inference by reducing computation requirements at each step via eliminating the need to recompute prior states. The authors achieved a notable 5.68-fold reduction in inference time and a 30% decrease in memory consumption compared to the Incremental Inference method (Romaniuk et al., 2020). STMC enables the conversion of a standard CNN model, which typically requires an input of size at least as large as its receptive field, into a model capable of processing a single input frame at a time, akin to Long-Short Term Memory networks (LSTM). The SOI method is built upon the STMC foundation, offering distinct treatment of strided layers and yielding a compelling new *inference pattern*[1].

Routing methods constitute a popular category of algorithms tailored to optimize the inference process of Recurrent Neural Networks (RNNs). In the field of Natural Language Processing (NLP), Yu et al. (2017) introduced an approach that involves traversing segments of the computational graph. This traversal is guided by decisions made by a reinforcement learning model following the processing of a fixed number of words. Another notable contribution by Campos et al. (2018) yielded an algorithm capable of selectively bypassing partial state updates within an RNN during inference, influenced by the input's length. In NLP terms, this concept can be compared to the mechanism of skipping words.

The research by Jernite et al. (2017) introduced a distinct method to regulate computation within a recurrent unit. This method relied on a scheduler unit that facilitated partial updates to the network's state, only when the computational budget limit was reached. Meanwhile, Seo et al. (2018) proposed an approach referred to as "word skimming". In this approach, the authors designed both small and large RNN models that could be interchangeably utilized for inference through the utilization of Gumbel softmax. The exploration of hybrid techniques combining jumping, skimming, and skipping was further advanced by Hansen et al. (2019), who published additional solutions in this direction.

The RNN routing methods have found successful applications in CNNs as well. Wang et al. (2018) introduced an approach that enables a model to learn a policy for skipping entire convolutional layers on a per-input basis. A similar concept, involving adaptive inference graphs conditioned on image inputs, was put forth by Veit & Belongie (2018). Additionally, several authors have contributed methods for early-exit during CNN inference (Bolukbasi et al., 2017; Teerapittayanon et al., 2016; Huang et al., 2017). In these methods, the network is trained to skip portions of the computational graph towards the final stages, based on the characteristics of the input.

Other commonly employed methods for model optimization include pruning (LeCun et al., 1989) and quantization (Gray & Neuhoff, 1998). Importantly, both of these methods are not mutually exclusive and can coexist alongside routing methods within a single neural network.

1.2 Novelty

In this study, we introduce a method for reducing the computational cost of a ANN model while incurring only a negligible decrease in the model's performance. Importantly, these reductions are achieved with minimal alterations to the architecture, making it suitable for various tasks where factors such as energy consumption or time are of paramount importance.

Our approach involves the conversion of a conventional ANN model, initially trained to process segments of time-series data with arbitrary lengths in an offline mode, into a model that processes

---

[1]By the term of an inference pattern we mean a full computational graph of a single inference or of repeating sequence of inferences if they influence the computational graphs of the following model executions as in case of SOI or STMC.

the data element by element, enabling real-time usage. Notably, our method builds upon the STMC technique (Stefański et al., 2023). STMC is designed to perform each distinct operation exactly once. SOI extends this concept by omitting some operations in a structured manner.

In this study, we introduce a novel method named Scattered Online Inference (SOI), which is based on the following key principles:

- The reduction of computational complexity is achieved through the implementation of partial predictions of the network's future state.
- SOI operates as a two-phase system. The initial phase involves compressing data within the time domain using data compression. The subsequent phase focuses on data reconstruction, employing the most suitable extrapolation scheme.
- The method preserves the causal nature of the optimized network architecture.
- SOI's applicability is confined to a single-frame online inference and it necessitates the incorporation of skip connections to update the network's output following each inference.

1.3 Limitations

While our method, demonstrates significant advantages in reducing computational complexity, it is essential to acknowledge several limitations inherent to our approach.

First, the performance reduction observed with SOI, although acceptable within the context of our work, may not be tolerable in applications requiring the highest accuracy levels. This reduction, while compensated by a substantial reduction in computational cost, suggests a trade-off between efficiency and model performance that may limit SOI's applicability in scenarios where performance is the absolute priority.

Second, the flexibility provided by SOI to balance computational cost and model performance introduces complexity in selecting the optimal configuration. The method allows users to adjust this trade-off based on application requirements, but this demands careful tuning and validation, which could be resource-intensive in training.

Third, the method's reliance on partial state predictions and data compression may introduce cumulative errors over time, particularly in longer sequences of time-series data. This is less critical in real-time, short-sequence applications but could degrade performance in tasks requiring continuous operation over extended periods without full model recalculations.

Lastly, SOI's applicability is primarily demonstrated on specific neural network architectures. Although we expect SOI to generalize to other architectures, its effectiveness in reducing computational complexity while maintaining acceptable performance may vary depending on the network design, the nature of the task, and the dataset. Further research and experimentation are necessary to explore its utility across a broader range of models and applications.
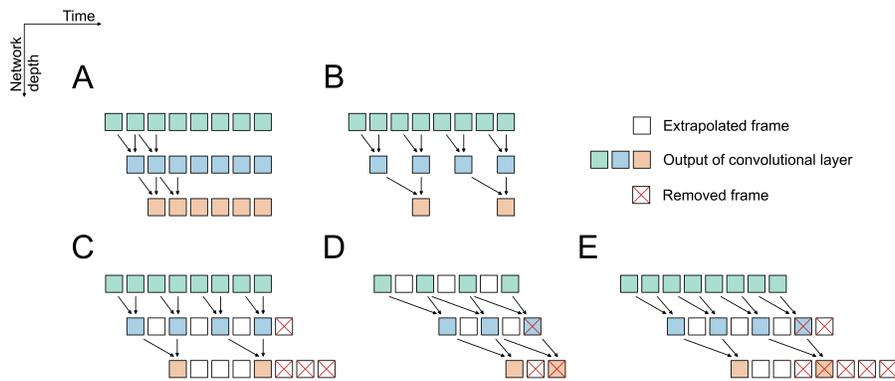

Figure 1: SOI for convolutional operations. For visualization purposes we show data as frames in time domain. A) Standard convolution. B) Strided convolution. C) Strided-Cloned Convolution. D) Shifted convolution. E) Shifted Strided-Cloned Convolution.
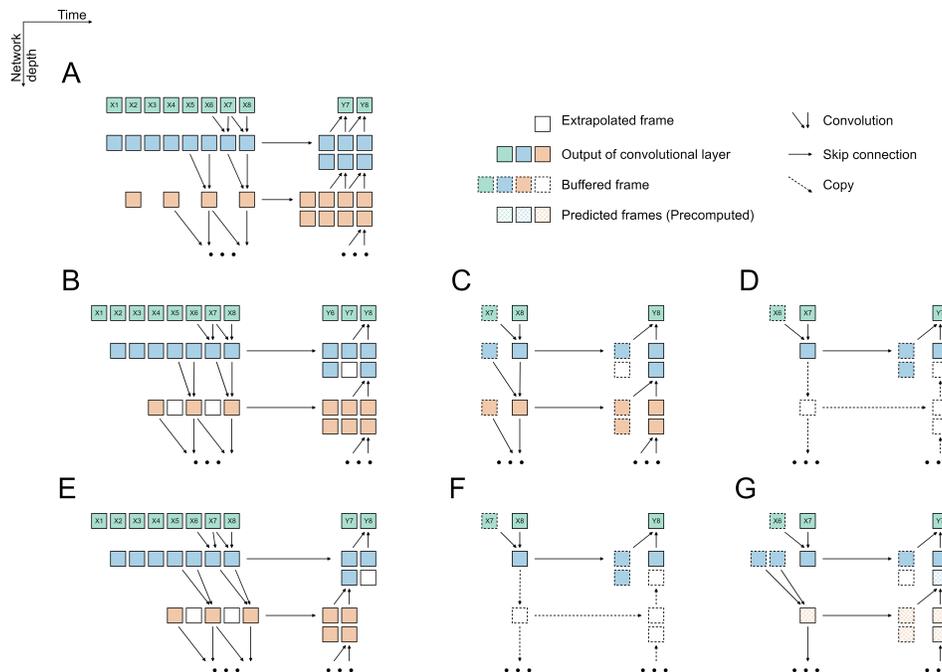
Figure 2: Inference patterns of each type of SOI based on U-Net architecture. A) Unmodified causal U-Net. B) Partially predictive (PP) SOI. C) Even inference of PP. D) Odd inference of PP. E) Fully predictive (FP) SOI. F) Even inference of FP. G) Odd inference of FP.

2 Methods

When processing a time-series in online mode, the model goes through each incoming element[2] separately. In this paper we refer to such an event as *inference*.

An improvement in computation complexity is achieved by introducing the partially predictive compression layer adapted for online inference, as well as by avoiding the redundant computations done during previous inferences as in STMC study. Therefore, after each inference, the results which would be recalculated in subsequent runs are cached. We refer to such cacheable outputs as a *partial state* of the network.

2.1 Scattered Online Inference

Scattered Online Inference (SOI) is a method which modifies the inference pattern of a network to skip the recalculation of certain layers in a predetermined pattern. This is achieved through the use of compression and extrapolation. Both operations are exclusively applied along the time axis. In this study, as example, we employed strided convolutions as compression layers and a basic form of extrapolation, where the last known value is used to replace the missing ones[3]. To facilitate a better understanding of the SOI algorithm in CNNs, in Figure 1, we define three types of convolutional layers utilized in our method. For comparison purposes, Figure 1 also includes standard convolution and strided convolution.

Strided-Cloned Convolution (S-CC) performs a 2-step operation. Firstly, it applies a strided convolution to the input in order to compress the data. In the second step it fills the gaps created by striding using any form of extrapolation. In our experiments we extrapolate by simply copying a previous frame. The copied frame is then aligned with a future frame relatively to its input. In practice, we split the stride and extrapolation operations into different layers which results in optimization of computational complexity between those layers. Because of that we will refer to this as S-CC pair.

---

[2]We define an "element" as a chunk of data of arbitrary size, including the smallest possible unit - a singular data point in a time series. The size of the incoming data chunk depends on the specific system being used.

[3]More sophisticated methods were tested as well and are presented in supplementary materials.

Shifted Convolution (SC) shifts data in time domain after the convolution thus creating additional predicted network states. This layer may be used for additional latency reduction.

Shifted Strided-Cloned Convolution (SS-CC) is a combination of S-CC pair and SC layer which is needed if we want to do both of them at the same point of the network. In our experiments we extrapolate output vector first and then apply data shifting to reduce size of introduced partial state prediction.

SOI can be divided into two types depending on how prediction is handled. These two types have significantly different inference patterns (Fig. 2).

Partially Predictive (PP) In this type of SOI, we do not introduce any additional predictive components in the network. This implies that after compression, the most recent frame stores information for the current output frame and the future output frame. This type of SOI uses S-CC pairs only. This configuration results in only one inference (the first one), which updates all network states, while all subsequent ones use relevant buffered partial network states. Although hybrid setups involving more full passes are viable, they fall outside the scope of this paper. It's important to note that this mode does not reduce peak computational complexity, but rather the average computational complexity.

Fully Predictive (FP) In this type of SOI, we introduce additional predictive components to the network. Compared to PP SOI, the most recent frame does not store any information about the current output frame, but rather about two future ones, hence the name "Fully Predictive", as only the future output is calculated. This is a more challenging task than PP SOI, but it can significantly decrease the latency of the model. This mode utilizes both S-CC pairs and SC layers. It optimizes both peak computational complexity and latency because it allows some inferences to be predicted in full. The fully predicted inference, in contrast to the regular inference which requires newly collected input, operates only on already processed data and can calculate relevant network states while the system awaits the new data, reducing the amount of computation required when the new data arrives.

Both of these types of SOI can be combined. This occurs when we first introduce PP SOI compression and then after some number of layers, we introduce an additional shift in the time axis after which the model can be treated as FP SOI.

2.2   Mathematical Formulation

Let us assume that an input of the model is represented by a 1D time series vector $X \in \mathbb{R}^N$ composed of N samples. Additionally let's say that a network is composed of 5 convolutional layers and output vector $^l Y$ for l-th layer is of the same shape as the input. Each convolutional layer has a 1D kernel $h_l \in \mathbb{R}^{M_l}$ which can be represented by a Toeplitz matrix $H_l \in \mathbb{R}^{N \times (N - M_l + 1)}$. We get:

$$^l Y = H_l \cdot {}^l X^T \tag{1}$$

After which we apply activation function σ and get the input for the next layer:

$$^{l+1} X = \sigma(^l Y) \tag{2}$$

We use $^l X_t$ to denote a segment of X ending in time t of a length matching the context it is used in (e.g. assuming h_l denotes a convolutional kernel of layer l and provided with the context $h_l \cdot {}^l X_t$, $^l X_t$ has M_l elements to match the kernel size).

By using the STMC inference pattern we perform inference for each element of X separately and reuse past partial states of each layer to fully reconstruct the receptive field of the network which can be represented as follows:

$$^{l+1} X_t = \left({}^{l+1} X_{t-1} \,\Big|_t\, \sigma(h_l \cdot {}^l X_t^T) \right) \tag{3}$$

where $(\cdot \,|_t\, \cdot)$ represents the concatenation of vectors in time axis.

If we use a convolutional layer with a stride of 2 as our second layer, then in the standard pattern, the size of the output vector of this layer is halved compared to its input. Consequently, every subsequent layer also has its input and output tensors reduced to half the size. We can restore the output to its original size by, for instance, applying transposed convolution. Let's assume that we apply transposed

convolution in the 4th layer of our network. In comparison to our initial plain network, the new strided network will have the same number of operations in both the first and last layers. The 2nd, 3rd, and 4th layers will each have half the computations as before. In the 2nd layer, this reduction is due to the application of stride. In the third layer, it is a result of the smaller input size. Similarly, in the 4th layer, if we disregard multiplications by zero.

When employing the STMC inference method, it's anticipated that each layer should process a single element and produce a single output. If we apply a similar stride and transposed convolution pattern without any additional modification, we'll face difficulties in reconstructing the output. Strided convolution would provide output values for even-numbered inferences (assuming a stride size of 2). However, during odd inferences, the 4th layer (transposed convolution) would require an upcoming even-numbered inference value, which would not yet be available. The authors of STMC propose a solution where every inference is treated as even-numbered, maintaining separate states for odd and even input positions. However, this pattern presents a challenge due to the exponential increase in the number of states for each added strided convolution.

SOI addresses this issue by removing the necessity of storing additional states, albeit at a cost to measured performance. For instance, in our network, we achieve this by abstaining from calculating the outputs of the 2nd, 3rd, and 4th layers during every even inference. To maintain causality, the transposed convolution output must be temporally shifted to produce either current and future frames or solely future frames, depending on the chosen SOI inference mode. Additionally, we advocate for the use of a skip connection between the input of the strided convolution and the output of the transposed convolution to update deeper layers of the network with information about the current data. This operation aims to minimize the influence of data forecasting on the optimized part of the network.

To formally describe partially predictive SOI, let's assume that the network contains layer l_d with a stride of size 2^4 and layer l_u which reverses the downsampling performed by l_d. Typically, l_d would be in the encoder part, while l_u would be in the decoder.

The downsampling layer only returns a new element for even-numbered inferences. It's important to note that until the upsampling layer is reached, there is no need to perform any further computations when t is odd. Namely for l such that $l_d < l \le l_u$:

$$
{}^{l+1}X_t = \begin{cases} \left({}^{l+1}X_{t-1} \,\Big|_t\, \sigma(h_l \cdot {}^lX_t^T) \right), & \text{if } t \text{ is even} \\ {}^{l+1}X_{t-1}, & \text{if } t \text{ is odd} \end{cases} \tag{4}
$$

At layer l_u we reconstruct the output by duplicating the convolution output.

$$
{}^{l_u+1}X'_t = \left({}^{l_u+1}X'_{t-1} \,\Big|_t\, \sigma(h_{l_u} \cdot {}^{l_u}X_t^T)^T \right) \tag{5}
$$

Note that ${}^{l_u}X_{2s}^T = {}^{l_u}X_{2s+1}^T$.

We then concatenate the output with data from the skip connection ($(\cdot \,|_c\, \cdot)$ represents the concatenation of vectors in channel axis).

$$
{}^{l_u+1}X_t = \left({}^{l_u+1}X'_t \,\Big|_c\, {}^{l_d}X_t \right) \tag{6}
$$

In the example above we traded inference over $l_u - l_d$ layers at the cost of inference over additional channels from skip connection concatenation. The additional cost of skip connection does not take effect in architectures where skip connections naturally exist like U-Net which we use as one of the examples in this study.

For fully predictive variant we modify equation (5) and add a shift in time axis.

$$
{}^{l_u+1}X'_t = \left({}^{l_u+1}X'_{t-1} \,\Big|_t\, \sigma(h_{l_u} \cdot {}^{l_u}X_{t-1}^T)^T \right) \tag{7}
$$

Figure 3: SOI PP inference pattern.

SOI can be seen as a form of forecasting where instead of training the model to predict output for input yet unknown, we predict partial states of the network. This is because preserving causality in
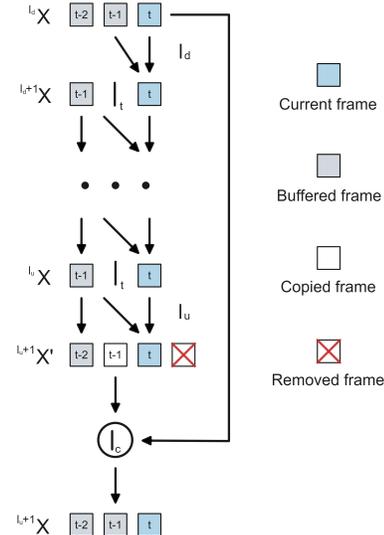
---

[4]The choice of 2 is for notational simplicity. The same derivation can be applied for an arbitrary stride size.

the strided convolution forces us to predict a convolution results when the next second input is not yet available.

3   Experiments

3.1   Speech Separation

We selected speech separation as our first experimental task. The choice of the task is dictated by our current research interests and the potential benefits of fast online inference. In this task we focus on separating clean speech signal from noisy background. In literature this task is also referred to as speech denoising or noise suppression.

For this experiment we adopted the U-Net architecture as it is widely used for this specific task and inherently has skip connections which will allow for applying SOI inference pattern without substantial alterations. Our model is composed of 7 encoder and 7 decoder layers. The Deep Noise Suppression (DNS) Challenge - Interspeech 2020 dataset (Reddy et al., 2020), licensed under CC-BY 4.0, was used for both training and evaluation purposes.

Position of S-CC pair   By introducing S-CC pair to the network we are enforcing data predictiveness of the network. The exact number of the predicted future partial states of the model depends on the position of S-CC pair and number of those pairs within the network. In addition it is worth noting that larger amount of predicted partial states of the network leads to higher reduction of computational complexity. In this experiment we test every position of S-CC pairs while applying up to two such modules to the model's architecture.

Position of SS-CC pair   SS-CC pair introduces additional shift in time axis compered to S-CC pair. In this experiment we alter the position of SS-CC pair within the network and also separately alter the position of S-CC pair and time shift which might be consider as a hybrid of partially and fully predictive pattern.

Resampling   Simple resampling of audio signal may be used to reduce the number of calculations done by neural networks but will yield significant increase in model latency. Nonetheless in this experiment we compare SOI to four different resampling methods: SoX which is based on method proposed by Soras (2004), using Kaiser window, polyphase and linear. With this methods we resampled our speech separation dataset from 16k to 8k at the input of the model and then from 8k to 16k at the output of the model. For every resampling we used our baseline model and compared it to three selected SOI models.

Pruning   In this experiment we used unstructured global magnitude pruning (similar to Han et al. (2015)) to show how our method can be combined with pruning leading to better results than pruning on a standard model. For this experiment we chose to prune "SOI 1" and "SOI 2|6" variants of our baseline model. Each step we pruned 4096 weights from model and we report how models performed on each step.

3.2   Acoustic Scene Classification

Acoustic scene classification (ASC) is our second task of choice. The goal of the task is to estimate the location where the specific sample was recorded. This task is commonly considered as an auxiliary problem in various online scenarios such as selection of bank of denoising filters.

For all our tests in ASC task we used GhostNet architecture (Han et al., 2020). We tested 7 different model sizes for all 3 variants - Baseline, STMC and SOI. Each test was repeated 5 times. We used the TAU Urban Acoustic Scene 2020 Mobile dataset (Heittola et al., 2020) for both training and validation. Models were trained on a single Nvidia P40 GPU for 500 epochs using Adam optimizer with initial learning rate of 1e-3.

4 Results

4.1 Speech Separation

Partially Predictive   Results of partially predictive SOI in speech separation task are shown in figure 4. We tested variants with a single S-CC layer ("S-CC") and two S-CC layers ("2xS-CC X"). In the latter case, the experiments are grouped by the position of the first S-CC layer (X). Value of SI-SNRi metric is highly dependant on the position of S-CC layers. Generally, the earlier the S-CC layer is introduced the lower SI-SNRi but higher complexity reduction. This phenomena may be explained by the difficulty of the partial state prediction task. All layers subsequent to the S-CC layer are required to predict twice the number of elements compared to the layers without having the S-CC as an prior layer. Overall, with SOI we can achieve linear dependency between computation cost and SI-SNRi up to 64% of complexity reduction (with linearity factor of 0.001 dB SI-SNRi per 1 MMAC/s (0.017 dB SI-SNRi per 1% of reduction)). A significant SI-SNRi drop is observed if S-CC layer is introduced too early. In table 1 we present selected results from the whole experiment.

Fully Predictive   Results for speech separation using fully predictive SOI are presented in figure 5. Details for selected models can be found in table 2. Reduction of metrics observed for fully predictive variant tends to be larger than for a single S-CC layer in partially predictive SOI. Additional reduction of metrics stems from added shift in time axis. Added shift produces a pattern where some part of model can be updated between inferences, as it only depends on past data. Size of this model part depends on the position of an SC or SS-CC layer within network. We refer to the size of this model part as "Precomputed" in table 2. The increase of precomputed partial state leads to reduction of SI-SNRi metric but allows to speed-up computation, because this part can be computed in advance. With FP SOI we achieved 50% reduction of computational cost at the expense of 11.3% of SI-SNRi but with 83.7% of network calculated using past data only.
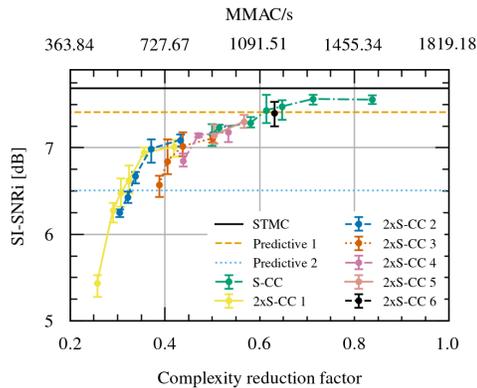

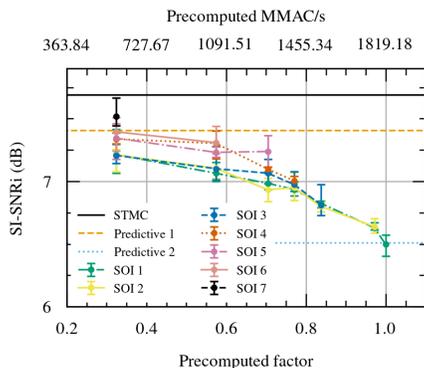Figure 4: Results of speech separation experiment with PP SOI.

Table 1: Selected results from experiments with partially predictive SOI for speech separation.

Model	SI-SNRi (dB)	SI-SNRi retain (%)	Complexity retain (%)	Complexity (MMAC/s)
STMC	**7.69** $^{+0.06}_{-0.08}$	**100.0**	100.0	1819.2
Predictive 1	7.41 $^{+0.07}_{-0.09}$	96.3	100.0	1819.2
Predictive 2	6.51 $^{+0.03}_{-0.08}$	84.7	100.0	1819.2
S-CC 2	7.23 $^{+0.04}_{-0.05}$	94.0	51.4	935.2
S-CC 5	7.47 $^{+0.07}_{-0.15}$	97.2	64.8	1178.7
S-CC 7	7.55 $^{+0.05}_{-0.05}$	98.2	83.8	1524.3
2xS-CC 1|3	6.27 $^{+0.09}_{-0.14}$	81.6	29.1	528.8
2xS-CC 1|6	6.94 $^{+0.07}_{-0.03}$	90.2	35.6	648.5
2xS-CC 2|5	6.67 $^{+0.05}_{-0.09}$	86.8	33.8	615.0
2xS-CC 3|6	7.02 $^{+0.16}_{-0.12}$	91.3	43.8	796.4
2xS-CC 4|6	7.14 $^{+0.02}_{-0.02}$	92.9	47.1	857.3
2xS-CC 5|7	7.30 $^{+0.08}_{-0.07}$	94.9	56.7	1031.2
2xS-CC 6|7	7.40 $^{+0.13}_{-0.15}$	96.2	63.2	1149.5

Figure 5: Results of speech separation experiment with FP SOI.

Table 2: Selected results from experiments with fully predictive SOI for speech separation.

Model	SI-SNRi (dB)	SI-SNRi retain (%)	Complexity retain (%)	Complexity (MMAC/s)	Precomputed (%)
STMC	**7.69** $^{+0.06}_{-0.08}$	**100.0**	100.0	1819.2	0.0
Predictive 1	7.41 $^{+0.07}_{-0.09}$	96.3	100.0	1819.2	**100.0**
Predictive 2	6.51 $^{+0.03}_{-0.08}$	84.7	100.0	1819.2	**100.0**
SS-CC 2	6.64 $^{+0.07}_{-0.05}$	86.3	51.4	935.2	97.2
SS-CC 5	7.24 $^{+0.13}_{-0.16}$	94.1	64.8	1178.7	70.4
SS-CC 7	7.52 $^{+0.15}_{-0.07}$	97.8	83.8	1524.3	32.4
S-CC 1|3	6.82 $^{+0.02}_{-0.04}$	88.7	**50.0**	**909.6**	83.7
S-CC 1|6	7.06 $^{+0.09}_{-0.06}$	91.8	**50.0**	**909.6**	57.4
S-CC 2|5	6.93 $^{+0.09}_{-0.09}$	90.1	51.4	935.2	70.4
S-CC 3|6	7.10 $^{+0.18}_{-0.09}$	92.3	58.1	1057.5	57.4
S-CC 4|6	7.30 $^{+0.10}_{-0.12}$	94.9	61.5	1118.4	57.4
S-CC 5|6	7.23 $^{+0.05}_{-0.04}$	94.0	64.8	1178.7	57.4
S-CC 6|7	7.39 $^{+0.12}_{-0.15}$	96.1	71.3	1296.9	32.4

19360

Resampling The results presented in the table 3 show that SOI outperforms the listed resampling methods in terms of complexity reduction under the constrained of preserving the quality of the original model (STMC). The results for resampling-based complexity reduction indicate that model's quality depends on the used resampling algorithm and, for all the selected ones, the initial information loss strongly affects the model performance.
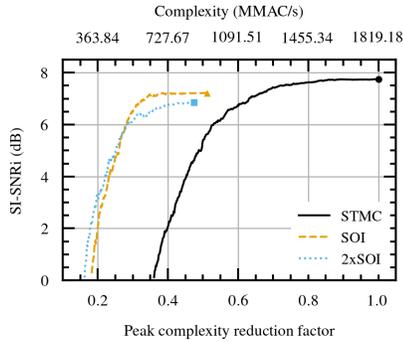

Figure 6: Pruning of STMC, SOI and 2xSOI models. Unpruned models are indicated by markers.

Table 3: Comparison between resampling and SOI.

Method	SI-SNRi (dB)	Complexity (MMAC/s)
STMC	7.69	1819.2
Linear	3.49	909.6
Polyphase	5.69	909.6
Kaiser	5.83	909.6
SoX	5.77	909.6
S-CC 5	**7.47**	1178.7
S-CC 2	7.23	935.2
S-CC 1|3	6.27	**528.8**

Pruning The application of SOI along with the pruning of STMC model surpasses the effect of application of the pruning alone. The addition of SOI allowed for achieving a further reduction in computational complexity by around 300 MMAC/s for the same model's performance, which is about 16% of the original model. Interestingly the "SOI 2\|6" surpassed the "SOI 1" model at around 6 dB SI-SNRi. The results of the experiment are shown in Fig. 6.

4.2 Acoustic Scene Classification

Results for the ASC task are collected in Table 4. For models I, III and VII we observed that our method led to an increase in accuracy, whereas other models showed a the decrease in metrics. The largest decrease in accuracy was around 2.20%, and the largest increase was 1.69%. These results indicate that SOI does not lead to a decrease in model quality for this particular task compared to the STMC model. This can be explained by the much slower change of output (acoustic scene label) compared to the previous task (speech mask). In our tests, the reduction in computational complexity of SOI models amounted to around 16% compared to the STMC model. This reduction decreased to 11% for the smallest model due to the addition of the skip connections. For this particular architecture, our method also reduced the number of parameters for the largest tested models by around 7%.

Table 4: Results of ASC experiment.

Model	Method	Top-1 Accuracy (%)	Complexity (MMAC/s)	Number of Parameters
I	Baseline	$55.68^{+0.62}_{-0.96}$	423.07	**1470**
	STMC	$55.68^{+0.62}_{-0.96}$	0.41	**1470**
	SOI	$\mathbf{55.90}^{+1.83}_{-5.61}$	**0.37**	1833
II	Baseline	$\mathbf{64.18}^{+0.72}_{-0.43}$	959.67	**3352**
	STMC	$\mathbf{64.18}^{+0.72}_{-0.43}$	0.94	**3352**
	SOI	$61.98^{+1.06}_{-1.38}$	**0.80**	3594
III	Baseline	$66.45^{+1.75}_{-1.69}$	1624.11	**5814**
	STMC	$66.45^{+1.75}_{-1.69}$	1.59	**5814**
	SOI	$\mathbf{68.14}^{+0.92}_{-2.23}$	**1.37**	6653
IV	Baseline	$\mathbf{70.57}^{+2.64}_{-1.81}$	2405.09	**8696**
	STMC	$\mathbf{70.57}^{+2.64}_{-1.81}$	2.35	**8696**
	SOI	$70.32^{+1.17}_{-1.98}$	**1.97**	8835
V	Baseline	$\mathbf{76.91}^{+0.74}_{-1.40}$	6769.78	25480
	STMC	$\mathbf{76.91}^{+0.74}_{-1.40}$	6.61	25480
	SOI	$76.42^{+0.66}_{-0.63}$	**5.54**	**24632**
VI	Baseline	$\mathbf{81.66}^{+1.00}_{-0.98}$	13187.40	50392
	STMC	$\mathbf{81.66}^{+1.00}_{-0.98}$	12.78	50392
	SOI	$80.73^{+1.79}_{-1.36}$	**10.75**	**47605**
VII	Baseline	$83.07^{+1.03}_{-0.68}$	21395.26	83432
	STMC	$83.07^{+1.03}_{-0.68}$	20.87	83432
	SOI	$\mathbf{83.35}^{+0.74}_{-1.12}$	**17.59**	**77753**

5 Conclusion

In this work, we presented a method for reducing the computational cost of a convolutional neural network by reusing network partial states from previous inferences, leading to a generalization of these states over longer time periods. We discussed the effect of partial state prediction that our method imposes on the neural model and demonstrated that it can be applied gradually to balance model quality metrics and computational cost.

Our experiments highlight the high potential for computational cost reduction of a CNN, especially for tasks where the output remains relatively constant, such as event detection or classification. We achieved a computational cost reduction of 50% without any drop in metrics in the ASC task and a 64.4% reduction in computational cost with a relatively small reduction of 9.8% in metrics for the speech separation task. We also showcased the ability of SOI to control the trade-off between model's quality and computational cost, allowing for resource- and requirement-aware tuning.

The presented method offers an alternative to the STMC solution for strided convolution. While SOI reduces network computational complexity at the expense of measured performance, STMC ensures that metrics are not reduced but at the cost of increased memory consumption at an exponential rate. SOI is akin to methods like network pruning, but unlike pruning, it does not require special sparse kernels for inference optimization. It is worth noting that these methods are not mutually exclusive, therefore, the STMC strided convolution handler, SOI, and pruning can coexist within a neural network to achieve the desired model performance.

References

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022. doi: 10.1126/science.ade9097.

Vijay Balasubramanian. Brain power. *Proceedings of the National Academy of Sciences*, 118(32): e2107022118, 2021. doi: 10.1073/pnas.2107022118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2107022118.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 527–536. JMLR.org, 2017.

Víctor Campos, Brendan Jou, Xavier Giró-i-Nieto, Jordi Torres, and Shih-Fu Chang. Skip RNN: learning to skip state updates in recurrent neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HkwVAXyCW.

Jiaming Gong, Wei Liu, Mengjie Pei, Chengchao Wu, and Liufei Guo. Resnet10: A lightweight residual network for remote sensing image classification. In *2022 14th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pp. 975–978, 2022. doi: 10.1109/ICMTMA54903.2022.00197.

R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6): 2325–2383, 1998. doi: 10.1109/18.720541.

Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 1135–1143, Cambridge, MA, USA, 2015. MIT Press.

Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Neural speed reading with structural-jump-LSTM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xf9jAqFQ.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. URL https://api.semanticscholar.org/CorpusID:206594692.

Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. TAU Urban Acoustic Scenes 2020 Mobile, Development dataset. Zenodo, February 2020. doi: 10.5281/zenodo.3819968. URL https://doi.org/10.5281/zenodo.3819968.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2017.

Yacine Jernite, Edouard Grave, Armand Joulin, and Tomas Mikolov. Variable computation in recurrent neural networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1LVSrcge.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Simon B. Laughlin, Rob R. de Ruyter van Steveninck, and John C. Anderson. The metabolic cost of neural information. *Nature Neuroscience*, 1(1):36–41, May 1998. ISSN 1546-1726. doi: 10.1038/236. URL https://doi.org/10.1038/236.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Charles E. Leiserson, Neil C. Thompson, Joel S. Emer, Bradley C. Kuszmaul, Butler W. Lampson, Daniel Sanchez, and Tao B. Schardl. There's plenty of room at the top: What will drive computer performance after moore's law? *Science*, 368(6495):eaam9744, 2020. doi: 10.1126/science.aam9744.

Gordon E Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86 (1):82–85, 1998.

Thai Son Nguyen, Sebastian Stueker, and Alexander H. Waibel. Super-human performance in online low-latency recognition of conversational speech. In *Interspeech*, 2020.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Chandan KA Reddy, Vishak Gopal, Ross Cutler, Ebrahim Beyrami, Roger Cheng, Harishchandra Dubey, Sergiy Matusevych, Robert Aichner, Ashkan Aazami, Sebastian Braun, et al. The interspeech 2020 deep noise suppression challenge: Datasets, subjective testing framework, and challenge results. In *Proc. Interspeech 2020*, 2020.

Michal Romaniuk, Piotr Masztalski, Karol Piaskowski, and Mateusz Matuszewski. Efficient Low-Latency Speech Enhancement with Mobile Audio Streaming Networks. In *Proc. Interspeech 2020*, pp. 3296–3300, 2020. doi: 10.21437/Interspeech.2020-2443.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, dec 2020. doi: 10.1038/s41586-020-03051-4. URL https://doi.org/10.1038%2Fs41586-020-03051-4.

Biswa Sengupta and Martin B. Stemmler. Power consumption during neuronal computation. *Proceedings of the IEEE*, 102(5):738–750, 2014. doi: 10.1109/JPROC.2014.2307755.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. Neural speed reading via skim-RNN. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sy-dQG-Rb.

Laurent Soras. The quest for the perfect resampler. 02 2004.

Grzegorz Stefański, Krzysztof Arendt, Paweł Daniluk, Bartłomiej Jasik, and Artur Szumaczuk. Short-term memory convolutions. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=4DU_HCijfJp`.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469, 2016. doi: 10.1109/ICPR.2016.7900006.

Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *European Conference on Computer Vision (ECCV)*, 2018.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*, volume 11217 of *Lecture Notes in Computer Science*, pp. 420–436. Springer, 2018. doi: 10.1007/978-3-030-01261-8_25. URL `https://doi.org/10.1007/978-3-030-01261-8_25`.

Xiaowei Xu, Yukun Ding, Sharon Xiaobo Hu, Michael Niemier, Jason Cong, Yu Hu, and Yiyu Shi. Scaling for edge inference of deep neural networks. *Nature Electronics 2018 1:4*, 1:216–222, 4 2018. ISSN 2520-1131. doi: 10.1038/s41928-018-0059-3. URL `https://www.nature.com/articles/s41928-018-0059-3`.

Adams Wei Yu, Hongrae Lee, and Quoc Le. Learning to skim text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1880–1890, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1172. URL `https://aclanthology.org/P17-1172`.

A   Reproducibility Notes

A.1   Speech Separation

For this experiment we adopted the U-Net architecture as it is widely used for this specific task and inherently has skip connections which will allow for applying SOI inference pattern without substantial alterations. Our model is composed of 7 encoder and 7 decoder layers, each comprising STMC/tSTMC, batch norm and ELU activation layers. Each model was trained for 100 epochs using Adam optimizer with initial learning rate of 1e-3. We trained each model on a single Nvidia P40 GPU 5 times and reported the average SI-SNRi. The mean training time of a single model amounted to about 14 hours. The Deep Noise Suppression (DNS) Challenge - Interspeech 2020 dataset (Reddy et al., 2020), licensed under CC-BY 4.0, was used for both training and evaluation purposes. For training, we used 16384 10s samples without any form of augmentation and for both validation and test sets we used 64 samples with similar setup to the training set.

A.2   Acoustic Scene Classification

For all our tests in ASC task we used GhostNet architecture (Han et al., 2020). Our baseline model is the original architecture with "same" padding making it not applicable in online scenario. STMC model changes the padding to manually-applied padding in left-most (oldest) side of data and applies STMC inference pattern. SOI model adds upsampling after each processing block and skip connections between downsampling/upsampling layers.

Models were trained on a single Nvidia P40 GPU for 500 epochs using Adam optimizer with initial learning rate of 1e-3. We tested 7 different model sizes for all 3 variants - Baseline, STMC and SOI. Each test was repeated 5 times. The mean training time of a single model amounted to about 4 hours. We used the TAU Urban Acoustic Scene 2020 Mobile dataset (Heittola et al., 2020) for both training and validation.

B   Strided Convolutions are Better for Longer Predictions

In this experiment, we investigated the impact of strided convolutions on predictive inference. Our test environment consisted of a U-Net model applied to a speech separation task. We examined two model variants: *Predictive* and *Strided Predictive*. The Predictive model serves as our baseline U-Net with an added time shift at the end. The Strided Predictive model, in addition to the time shift, incorporates strided convolutions in place of standard ones. For each model, we conducted tests with four different lengths of prediction, ranging from 1 frame to 4 frames. We performed five training runs for each model using the setup outlined in Section 3.1. The results of this experiment are displayed in Figure 7 and Table 5.
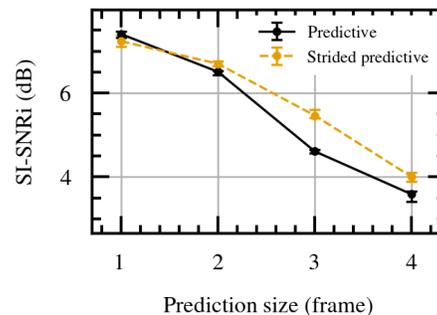

Figure 7: Comparison between standard convolutions and strided convolution in predictive inference.

We conclude that strided convolutions shows higher potential for longer predictions. We attribute this effect to the fact, that using strides forces stronger generalization of outputs of strided convolutions because they are applied in multiple contexts.

Table 5: Results of experiment on the influence of strided convolution on predictive inference.

Length of prediction	SI-SNRi (dB)	
	Predictive	Strided predictive
1	**7.41**$^{+0.07}_{-0.09}$	7.24$^{+0.14}_{-0.13}$
2	6.51$^{+0.03}_{-0.08}$	**6.70**$^{+0.07}_{-0.05}$
3	4.61$^{+0.04}_{-0.05}$	**5.47**$^{+0.14}_{-0.07}$
4	3.59$^{+0.08}_{-0.17}$	**4.00**$^{+0.11}_{-0.11}$

C   Inference Time and Peak Memory Footprint

To showcase SOI's influence on inference time and peak memory consumption, we extended our results from Table 1 to include these measurements with a single S-CC layer. The collected results are presented in Table 6.

Table 6: Results from experiments with partially predictive SOI for speech separation with added average inference time and peak memory footprint.

Model	SI-SNRi (dB)	SI-SNRi retain (%)	Complexity retain (%)	Complexity (MMAC/s)	Avg. inference time (ms)	Peak memory footprint (MB)
STMC	$\mathbf{7.69^{+0.06}_{-0.08}}$	**100.0**	100.0	1819.2	9.93 ± 0.13	27.2
Predictive 1	$7.41^{+0.07}_{-0.09}$	96.3	100.0	1819.2	9.93 ± 0.13	27.2
Predictive 2	$6.51^{+0.03}_{-0.08}$	84.7	100.0	1819.2	9.93 ± 0.13	27.2
S-CC 1	$7.15^{+0.12}_{-0.13}$	93.0	**50.1**	**911.4**	$\mathbf{5.28 \pm 0.07}$	**14.6**
S-CC 2	$7.23^{+0.04}_{-0.05}$	94.0	51.4	935.2	5.63 ± 0.11	18.7
S-CC 3	$7.28^{+0.07}_{-0.05}$	94.7	58.1	1057.5	6.27 ± 0.10	24.0
S-CC 4	$7.43^{+0.18}_{-0.14}$	96.7	61.5	1118.3	6.67 ± 0.13	25.1
S-CC 5	$7.47^{+0.07}_{-0.15}$	97.2	64.8	1178.7	6.98 ± 0.14	25.6
S-CC 6	$7.56^{+0.05}_{-0.06}$	98.3	71.3	1296.9	7.50 ± 0.08	26.1
S-CC 7	$7.55^{+0.05}_{-0.05}$	98.2	83.8	1524.3	8.43 ± 0.12	26.6

Additionally, in Figure 8, we show how inference time and peak memory consumption depend on SI-SNRi and the complexity reduction factor.
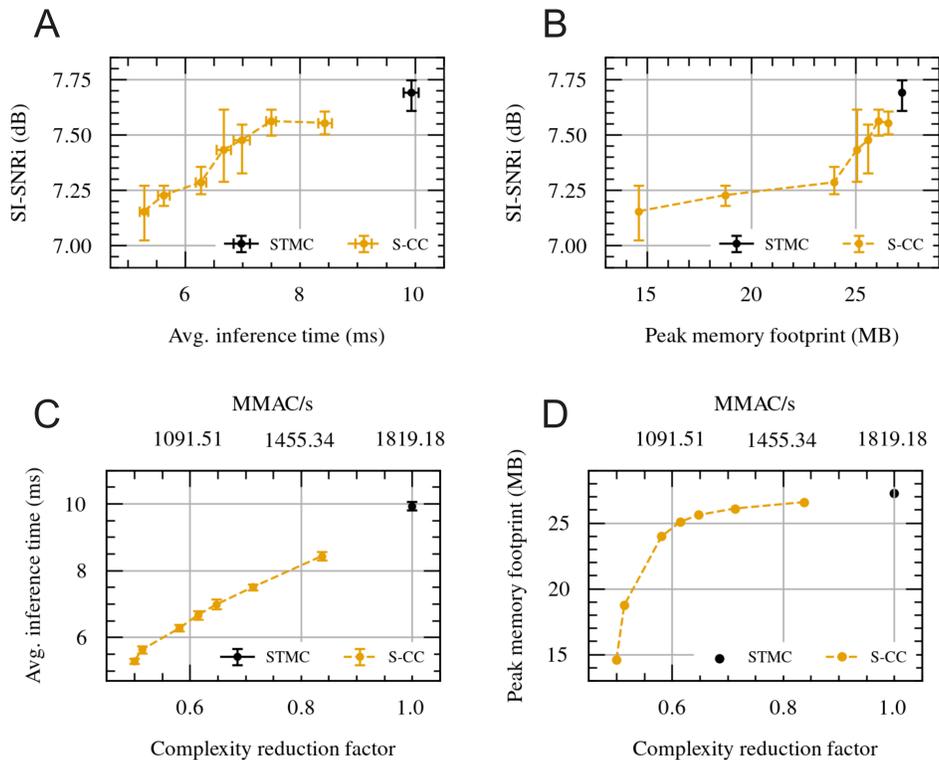

Figure 8: Average inference time and peak memory footprint

These results suggest that the inference time of the tested model depends linearly on SI-SNRi and the complexity reduction factor, while peak memory consumption has significant impact on SI-SNRi at the beginning. From 60% complexity reduction onward, the peak memory footprint decreases sharply.

D Interpolation

Our method may also benefit from the usage of interpolation methods in place of extrapolation. We tested a singular S-CC layer with three different types of interpolation: nearest-neighbor, bilinear, and bicubic. The results of this experiment can be observed in Figure 9 and Table 7. For comparison, we included our extrapolated duplication method in the results of this experiment.

In this experiment, we achieved the best results with bilinear or bicubic interpolation, although the results for bilinear interpolation showed much higher variance than any other method. It is important to remember that even though we achieve slightly better results using interpolation, the usage of interpolation comes at the cost of higher latency as we need to wait for an additional time frame.
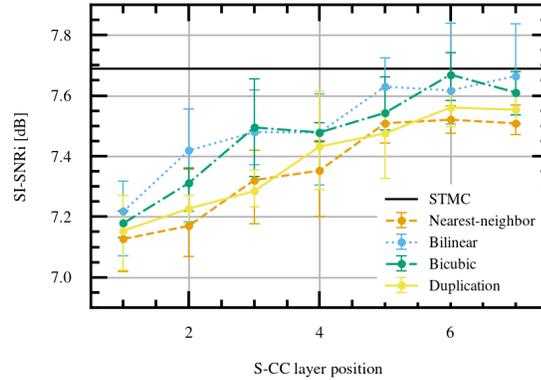

Figure 9: Comparison between extrapolated frame duplication and interpolation methods for PP SOI.

Table 7: Results of experiment on interpolation methods with PP SOI.

| Model | SI-SNRi (dB) | | | |
	Duplication	Nearest-neighbor	Bilinear	Bicubic
S-CC 1	$7.15^{+0.12}_{-0.13}$	$7.13^{+0.10}_{-0.11}$	$\mathbf{7.22}^{+0.11}_{-0.15}$	$7.18^{+0.05}_{-0.05}$
S-CC 2	$7.23^{+0.19}_{-0.11}$	$7.17^{+0.06}_{-0.05}$	$\mathbf{7.42}^{+0.14}_{-0.24}$	$7.31^{+0.05}_{-0.10}$
S-CC 3	$7.28^{+0.07}_{-0.05}$	$7.32^{+0.10}_{-0.15}$	$7.48^{+0.14}_{-0.11}$	$\mathbf{7.49}^{+0.17}_{-0.17}$
S-CC 4	$7.43^{+0.18}_{-0.14}$	$7.35^{+0.11}_{-0.16}$	$\mathbf{7.48}^{+0.13}_{-0.18}$	$\mathbf{7.48}^{+0.04}_{-0.04}$
S-CC 5	$7.47^{+0.07}_{-0.15}$	$7.51^{+0.12}_{-0.07}$	$\mathbf{7.63}^{+0.10}_{-0.13}$	$7.54^{+0.12}_{-0.06}$
S-CC 6	$7.56^{+0.05}_{-0.06}$	$7.52^{+0.05}_{-0.05}$	$7.62^{+0.23}_{-0.12}$	$\mathbf{7.67}^{+0.08}_{-0.09}$
S-CC 7	$7.55^{+0.05}_{-0.05}$	$7.51^{+0.07}_{-0.04}$	$\mathbf{7.66}^{+0.18}_{-0.16}$	$7.61^{+0.07}_{-0.08}$

E Different Extrapolation Methods

In the main paper we used element duplication as an extrapolation method but we also pointed out that any type of extrapolation may be used. Here we compare the results of our speech separation experiment using element duplication with another commonly found method – a transposed convolution. We tested both partially predictive and fully predictive modes using the setup described in Section 3.1.

Results of the experiment with PP SOI are presented in figure 10. In this case we tested a U-Net with two S-CC layers. Each plot represents a different position of the first S-CC pair and each point on X-axis represents a different position of the second S-CC pair. We also present results for hybrid models where duplication and transposed convolutions were used together – in the first and second S-CC pairs respectively.

In this experiment neither method demonstrated a significant advantage. It seems that duplication tends to perform better than transposed convolution if it is introduced deeper within the network and *vice versa* although difference is marginal.
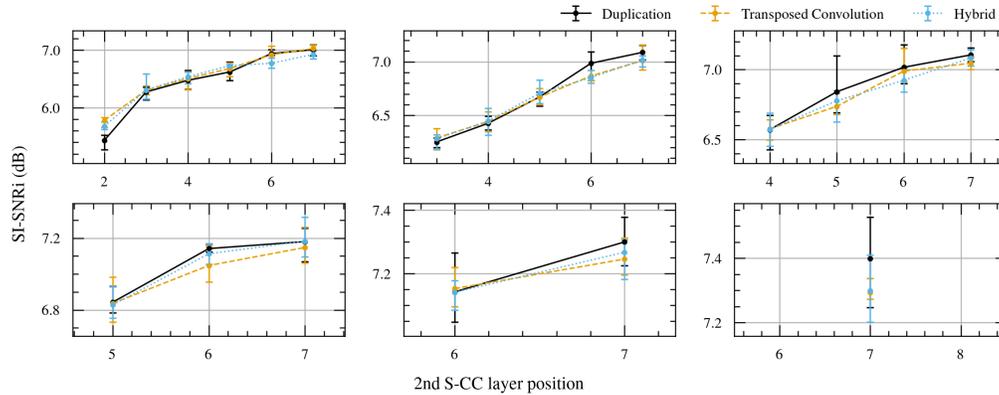
Figure 10: Comparison between frame duplication and transposed convolution for PP SOI.

Results achieved with FP SOI are shown in figure 11. Each plot represents a different position of S-CC pair and each point represents different position of SC layer. Achieved results confirm previous conclusions.

This experiment proves that element duplication is a viable method and thus should be chosen for its simplicity.

Table 8: Results of experiment on different extrapolation methods with PP SOI.

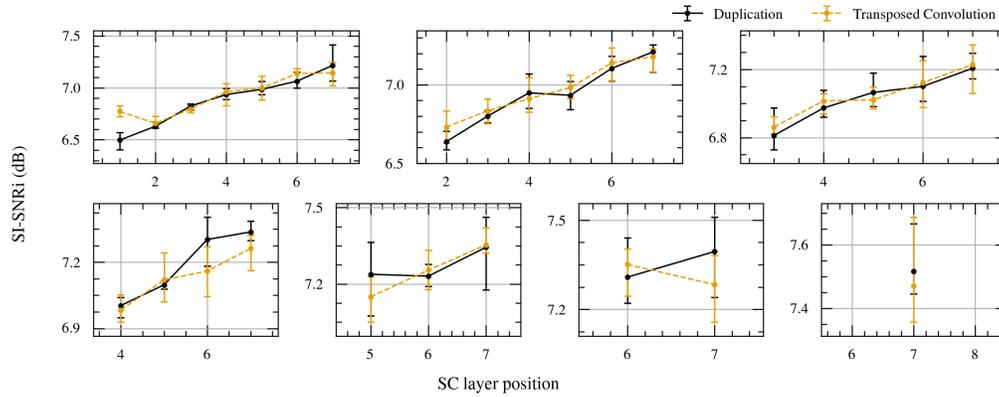| Model	SI-SNRi (dB)		
	Duplication	Transposed convolution	Hybrid
S-CC 1|2	$5.43^{+0.07}_{-0.17}$	$\mathbf{5.78}^{+0.05}_{-0.12}$	$5.68^{+0.08}_{-0.06}$
S-CC 1|3	$6.27^{+0.18}_{-0.14}$	$6.30^{+0.09}_{-0.08}$	$\mathbf{6.31}^{+0.29}_{-0.16}$
S-CC 1|4	$6.48^{+0.18}_{-0.16}$	$6.51^{+0.13}_{-0.20}$	$\mathbf{6.54}^{+0.08}_{-0.11}$
S-CC 1|5	$6.62^{+0.09}_{-0.15}$	$6.68^{+0.11}_{-0.14}$	$\mathbf{6.73}^{+0.03}_{-0.03}$
S-CC 1|6	$\mathbf{6.94}^{+0.10}_{-0.04}$	$6.92^{+0.15}_{-0.14}$	$6.77^{+0.10}_{-0.09}$
S-CC 1|7	$7.01^{+0.09}_{-0.11}$	$\mathbf{7.03}^{+0.07}_{-0.03}$	$6.92^{+0.07}_{-0.08}$
S-CC 2|3	$6.25^{+0.04}_{-0.06}$	$\mathbf{6.29}^{+0.09}_{-0.11}$	$6.28^{+0.04}_{-0.10}$
S-CC 2|4	$6.43^{+0.07}_{-0.07}$	$6.44^{+0.10}_{-0.10}$	$\mathbf{6.45}^{+0.12}_{-0.13}$
S-CC 2|5	$6.67^{+0.05}_{-0.09}$	$6.67^{+0.09}_{-0.07}$	$\mathbf{6.71}^{+0.13}_{-0.10}$
S-CC 2|6	$\mathbf{6.99}^{+0.11}_{-0.16}$	$6.87^{+0.06}_{-0.05}$	$6.85^{+0.07}_{-0.06}$
S-CC 2|7	$\mathbf{7.09}^{+0.07}_{-0.07}$	$7.01^{+0.15}_{-0.09}$	$7.02^{+0.05}_{-0.07}$
S-CC 3|4	$6.57^{+0.11}_{-0.15}$	$\mathbf{6.58}^{+0.07}_{-0.08}$	$6.57^{+0.12}_{-0.12}$
S-CC 3|5	$\mathbf{6.84}^{+0.26}_{-0.15}$	$6.74^{+0.10}_{-0.06}$	$6.78^{+0.08}_{-0.15}$
S-CC 3|6	$\mathbf{7.02}^{+0.17}_{-0.12}$	$6.99^{+0.17}_{-0.08}$	$6.92^{+0.09}_{-0.09}$
S-CC 3|7	$\mathbf{7.10}^{+0.05}_{-0.05}$	$7.05^{+0.05}_{-0.05}$	$7.08^{+0.07}_{-0.06}$
S-CC 4|5	$\mathbf{6.84}^{+0.09}_{-0.06}$	$\mathbf{6.84}^{+0.15}_{-0.11}$	$6.83^{+0.11}_{-0.08}$
S-CC 4|6	$\mathbf{7.14}^{+0.03}_{-0.18}$	$7.05^{+0.11}_{-0.10}$	$7.11^{+0.06}_{-0.06}$
S-CC 4|7	$\mathbf{7.18}^{+0.08}_{-0.12}$	$7.15^{+0.12}_{-0.09}$	$\mathbf{7.18}^{+0.14}_{-0.09}$
S-CC 5|6	$7.14^{+0.13}_{-0.10}$	$\mathbf{7.15}^{+0.07}_{-0.06}$	$7.14^{+0.04}_{-0.06}$
S-CC 5|7	$\mathbf{7.30}^{+0.08}_{-0.08}$	$7.25^{+0.07}_{-0.06}$	$7.27^{+0.04}_{-0.09}$
S-CC 6|7	$\mathbf{7.34}^{+0.13}_{-0.16}$	$7.29^{+0.05}_{-0.03}$	$7.30^{+0.12}_{-0.10}$

Figure 11: Comparison between frame duplication and transposed convolution for FP SOI.

Table 9: Results of experiment on different extrapolation methods with FP SOI.

Model	SI-SNRi (dB)	
	Duplication	Transposed convolution
SS-CC 1	$6.50^{+0.08}_{-0.10}$	$\mathbf{6.77}^{+0.06}_{-0.05}$
S-CC 1|2	$6.63^{+0.05}_{-0.02}$	$\mathbf{6.66}^{+0.07}_{-0.04}$
S-CC 1|3	$\mathbf{6.82}^{+0.03}_{-0.04}$	$6.80^{+0.04}_{-0.04}$
S-CC 1|4	$6.93^{+0.07}_{-0.05}$	$\mathbf{6.96}^{+0.08}_{-0.14}$
S-CC 1|5	$6.98^{+0.08}_{-0.06}$	$\mathbf{7.00}^{+0.12}_{-0.12}$
S-CC 1|6	$7.06^{+0.09}_{-0.07}$	$\mathbf{7.14}^{+0.05}_{-0.08}$
S-CC 1|7	$\mathbf{7.21}^{+0.20}_{-0.15}$	$7.14^{+0.11}_{-0.14}$
SS-CC 2	$6.64^{+0.05}_{-0.13}$	$\mathbf{6.73}^{+0.11}_{-0.11}$
S-CC 2|3	$6.80^{+0.08}_{-0.08}$	$\mathbf{6.83}^{+0.08}_{-0.09}$
S-CC 2|4	$\mathbf{6.95}^{+0.10}_{-0.10}$	$6.91^{+0.14}_{-0.09}$
S-CC 2|5	$6.93^{+0.13}_{-0.10}$	$\mathbf{6.98}^{+0.08}_{-0.07}$
S-CC 2|6	$7.10^{+0.04}_{-0.05}$	$\mathbf{7.14}^{+0.10}_{-0.12}$
S-CC 2|7	$\mathbf{7.21}^{+0.07}_{-0.06}$	$7.18^{+0.06}_{-0.10}$
SS-CC 3	$6.81^{+0.17}_{-0.09}$	$\mathbf{6.86}^{+0.07}_{-0.04}$
S-CC 3|4	$6.97^{+0.11}_{-0.06}$	$\mathbf{7.01}^{+0.05}_{-0.08}$
S-CC 3|5	$\mathbf{7.06}^{+0.12}_{-0.09}$	$7.02^{+0.08}_{-0.06}$
S-CC 3|6	$7.10^{+0.18}_{-0.09}$	$\mathbf{7.12}^{+0.13}_{-0.15}$
S-CC 3|7	$7.21^{+0.09}_{-0.07}$	$\mathbf{7.23}^{+0.12}_{-0.17}$
SS-CC 4	$\mathbf{7.00}^{+0.04}_{-0.06}$	$6.98^{+0.08}_{-0.06}$
S-CC 4|5	$7.10^{+0.02}_{-0.02}$	$\mathbf{7.12}^{+0.13}_{-0.10}$
S-CC 4|6	$\mathbf{7.30}^{+0.11}_{-0.13}$	$7.16^{+0.12}_{-0.12}$
S-CC 4|7	$\mathbf{7.34}^{+0.05}_{-0.04}$	$7.26^{+0.07}_{-0.10}$
SS-CC 5	$\mathbf{7.24}^{+0.13}_{-0.17}$	$7.15^{+0.08}_{-0.10}$
S-CC 5|6	$7.23^{+0.05}_{-0.04}$	$\mathbf{7.25}^{+0.08}_{-0.08}$
S-CC 5|7	$7.34^{+0.12}_{-0.17}$	$\mathbf{7.35}^{+0.07}_{-0.04}$
SS-CC 6	$7.31^{+0.14}_{-0.09}$	$\mathbf{7.35}^{+0.06}_{-0.11}$
S-CC 6|7	$\mathbf{7.39}^{+0.12}_{-0.16}$	$7.28^{+0.10}_{-0.13}$
SS-CC 7	$\mathbf{7.52}^{+0.16}_{-0.08}$	$7.47^{+0.22}_{-0.12}$

19369

https://doi.org/10.52202/079017-0611

F Video Action Recognition

Other domains can benefit from SOI, as it can be applied to any time-series data. To illustrate this, we conducted experiments using SOI for an action recognition task with video data. We utilized the HMDB-51 dataset Kuehne et al. (2011), which contains 24 fps video data of human actions split into 51 classes. We trained a popular ResNet-10 architecture Gong et al. (2022) in three variants: regular, small (with halved number of channels) and tiny (with number of channels reduced fourfold), where we replaced 3D convolutional layers with 3D STMC layers. Here, we applied SOI by introducing a skip connection between the output of block 2 and the input of block 4, optimizing block 3.

Table 10: Results of video action recognition experiment.

Model	Regular		SOI	
	Top-1 Acc (%)	Complexity (GMAC/s)	Top-1 Acc (%)	Complexity (GMAC/s)
ResNet-10	32.63	48.54	33.34	40.69
ResNet-10 small	31.24	15.05	31.41	13.09
ResNet-10 tiny	30.46	5.23	30.90	**4.73**
MoViNet A0	34.40	33.15	31.88	24.26
MoViNet A1	**35.96**	69.77	32.73	53.92

To demonstrate that SOI can work not only with STMC, we also applied it to MoViNets, which use a method called "Stream buffers". We trained two variants of MoViNets, A0 and A1, in their streaming form. SOI was applied by optimizing blocks 4 and 5. Please note that SOI can be used not only with 3D convolutions but also with their popular 2D+1 variant.

All models were trained for 100 epochs with Adam optimizer and learning rate 5e-5. Each model was trained on Nvidia A100 GPU with batch size of 16. The mean training time amounted to about about 37h.

Achieved results suggest that SOI is suitable for video domain as well. ResNet-10 architecture proved to be highly compatible with SOI for this task as results achieved with SOI variant of this model outperformed the regular ones. We believe that this improvement was imposed by the increase in receptive field as SOI adds additional strided convolution. The achieved reduction of complexity for this family of models was between 10-17% depending on model size. For MoViNets, the decrease of around 3% in accuracy can be spotted, although the imposed complexity reduction was higher compared to ResNet and amounted to 23-30%.

G Acoustic Scene Classification with ResNet

To further evaluate the effectiveness of our method, we conducted experiments on the accoustic scene classification task using ResNet architecture He et al. (2015). We chose ResNet for two primary reasons: (1) its widespread use and recognition as a standard deep learning model, and (2) its relatively large size compared to other architectures tested, such as GhostNet, which allows us to better assess how SOI performs on larger models.

We tested 4 different ResNet models for all 3 variants - Baseline, STMC and SOI. Each test was repeated 5 times. We used the TAU Urban Acoustic Scene 2020 Mobile dataset (Heittola et al., 2020) for both training and validation.

The results are presented in Table 11.

Table 11: Results of ASC experiment with ResNet.

Model	Method	Top-1 Accuracy (%)	Complexity (GMAC/s)	Number of Parameters
18	Baseline	$85.13^{+0.83}_{-1.32}$	143.65	
	STMC	$85.13^{+0.83}_{-1.32}$	15.56	11.7M
	SOI	$\mathbf{91.55^{+0.72}_{-0.86}}$	**12.35**	
34	Baseline	$86.03^{+1.36}_{-3.08}$	686.96	
	STMC	$86.03^{+1.36}_{-3.08}$	32.65	21.8M
	SOI	$\mathbf{92.01^{+0.69}_{-0.89}}$	**26.46**	
50	Baseline	$89.66^{+0.60}_{-0.54}$	794.34	
	STMC	$89.66^{+0.60}_{-0.54}$	33.10	25.6M
	SOI	$\mathbf{91.43^{+0.83}_{-1.89}}$	**27.99**	
101	Baseline	$94.74^{+0.40}_{-0.89}$	2168.81	
	STMC	$94.74^{+0.40}_{-0.89}$	112.84	44.5M
	SOI	$\mathbf{96.22^{+0.92}_{-0.95}}$	**95.83**	

The results, as shown in Table 11, demonstrate that the SOI-enhanced ResNets consistently outperformed the baseline models in terms of accuracy. Specifically, the SOI-based models achieved higher classification accuracy across all tested configurations. We belived this improvement can be attributed to two key factors:

- The introduction of SOI allows the model to generalize over longer time frames by expanding its receptive field. This helps the model capture more temporal context, which is crucial for tasks like ASC.
- SOI encourages the model to generalize its output states by predicting future partial states, which improves the model's ability to handle variations in the data.

Despite ResNet's depth and complexity, the method proved successful, not only reducing computational cost but also improving performance.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the claims made, including the contributions made in the paper and important assumptions and limitations.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations of proposed method are discussed in section 1.3.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Reproducibility notes are present in section A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We cannot open-source our codebase due to its commercial purpose.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please see section A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Results are accompanied by error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Please see section A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper presents a general method for reduction of computational complexity of neural networks.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Datasets and their accompanying licenses are listed in section A.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.