No Frame Left Behind

Repetitive Action Counting with State Space Transformers

Alec Kerrigan
University of Central Florida
Orlando, Florida
aleckerrigan@knights.ucf.edu

Connor Malley
University of Central Florida
Orlando, Florida
malleyconnor@knights.ucf.edu

Moazam Soomro
University of Central Florida
Orlando, Florida
Moazamsoomro@knights.ucf.edu

Fatemah Najafali
University of Central Florida
Orlando, Florida
fsnajafali@knights.ucf.edu

Abstract

Repetitive activity counting is an interesting sub-field of action recognition concerned with identifying each periodic action in a video. Existing solution greatly vary in approach. However, nearly every method relies on multiple forward passes of the same video. These passes either consist of non-overlapping consecutive windows or feeding the video at different sample rates. This is resultant from the computational difficulty of using large numbers of frames. This results in both slower performance, as well as the inability to understand longer periodic actions. We present an approach using the newly proposed Structured State Spaces to systematically reduce the feature dimension of video frames while retaining both short term and long term temporal information, critical to repetitive action counting. We show that our system is competitive with current state of the art approaches, and is able to count videos at frame rates not possible by other methods.

1. Introduction

Visual repetitive motion that happen in a occur in the wild like jumping with a rope, bouncing a ball, or clapping the hands are known as repetitive actions that occur in parallel. These periodic actions occur on a regular basis and can be observed in both natural and urban environments as part of our daily activities. It is humanly impossible and challenging to count repetitions of actions, especially those which are which have a small period length.

Repetitive counting is common within the computer vision applications such as counting the number of the same pattern activities in motion. The main goal is to classify and count the repeated actions in a video for a single sample within a period of time.

Predicting the repetitions in an action video is not an easy task as it some challenges associated with it: having only a few repetitions within a video, the period length of an action can vary drastically for example a video of chopping an onion compared to a video of earth rotating, variance in viewpoint and non-stationary repetition. There have been multiple ways this problem has been addressed in which are discussed in detail in 2. Our work initially starts by processing the repetitive video by computing the sample rate for the continuous motion in the video frames. A higher sample rates results in information loss and there are fewer frames for the repetitions. Hence, not much can be counted with fewer repetitions. To overcome this, both the RepNet model [5] and the sight and sound network [18] was implemented to generate counts for repeated actions with uniform sampling. The approach for each of them has video clips as input with various components as part of the architecture and finally the period predictor as output resulting in the number of counts for repetitive actions. However, the method during inference uses multiple passes for the video as the camera augmentation could differ based on scaling, rotation, or translation for each frame and chooses the sample rate resulting in the highest score counting prediction.

As previous methods require multiple passes to overcome the sample rate. Our proposed method is to interpolate the features of a video sample in one pass. The features are extracted from snippets of the video and passed through a ResNet-50 encoder to produce per-frame embeddings. These embeddings are passed through the temporal self-similarity matrix to compute the similarities between all the pairs of the embeddings which eventually would pre-

dict the period count of a video. However, with this model the loss rarely converged with training.

In our work, we pass the video through our model only once and are able to do this with the aid of state space sequence S4 block [9]. The S4 block learns temporal information by using continuous state space, long range dependencies, and a combination of fast discrete representations where the number of features is reduced from 2048 up to 256. The main backbone is similar to that of the RepNet model [5] with the period predictor and temporal self-similarity matrix to compute the final counts of the repetitive motions using the Countix dataset. The best results for our model produced 0.46 OBO and 0.47 MAE with 180 frames.

2. Related Work

RepNet Model. The RepNet model [5] produces features for each video frame in the network. We implemented two main components of the architecture: Temporal Self-similarity Matrix (TSM) and Period Predictor. TSMs are useful for human action recognition that basically matches similar pairs of period segments by forming a single channel that provides regularization and the identical pairs are computed using a similarity function. The period predictor is the final module of the model that generates two outputs, the period length estimation, and periodicity classification. Each of the outputs are produced from the final self-similarity matrix that has the representation of per frame and fully connected layers.

Sight and Sound. The use of repetitive counting is implemented using video clips with audio data as mentioned in [18]. Although, the final counting prediction is produced from various components, such as, temporal stride decision and reliability estimation. However, multiple passes for each video frame are used to produce the final output. We proposed using S4 decoder blocks and transformer layers to generate single passes for the period length estimation. In addition, we incorporated the Countix dataset that includes various activities of repeated actions as the video data. With this, the dataset was not fully complete because some videos were missing while some were broken, cleaning up the dataset about 500 were lost from the 8757 videos which was not a hindrance to our implementation.

Counting in Computer Vision. In the field of computer vision, counting objects in an image or video is a common task, a few examples are, feature extraction with regression from an image [13], using crowd densities [10], and cell counting [12]. Furthermore, the repetitive counting trend has been popular in recent years, particularly in videos. The video with repeated motion is displayed, and the period cycle length is computed to determine the number of repetitions with less usage of the TSM.

Real-World Repetition Estimation The authors [14] addressed repetition estimation by using wavelet transform to compensate non-stationary video, and time varying flow and its differentials to estimate repetitions. [14] modeled the 3D natural repetitive motion as expansion, rotation or translation, depending upon the curl of the flow field and divergence. They present an analysis of time varying flow gradient and derive three motion types in 2D field: oscillatory, constant and intermittent. To map the 3D motion view in 2D, two viewpoints were used, and derived eighteen different examples of repetitive view by applying differential operators on the flow field of 3D motion. This work was focused on frequency domain analysis or pattern matching, our work is does not use such such techniques and rather provides an end-to-end solution.

3. State Space Transformers

3.1. Definition

Structured State Spaces, introduced by [7] offers a novel way of modeling long term, as well as short term, dependencies of very long sequences in a far more efficient way than self-attention. Typically, all previous methods, such as RNNs, CNNs, and Transformers have produced interesting and unique modifications to handle long sequences such as Lipshitz RNNs [6] or Performers [2], they often still fail as the number of steps grow very large.

S4 sequence models solves this computation bottleneck by abstracting learning a set of state dimension matrices, then decomposing them in such a way as to efficiently model the change in information over time. This can be described by the following equation.

$$\begin{aligned} x^{'}(t) &= \boldsymbol{A}x(t) + \boldsymbol{B}u(t) \\ y^{'}(t) &= \boldsymbol{C}x(t) + \boldsymbol{D}u(t) \end{aligned} \tag{1}$$

Essentially, the next state $x^{'}$ is described by some state xt and an input signal $u(t)$ parameterized by learnable matrices A and B, while the following output signal is described by the same state and input signal, but parameterized by matrices C and D. [7] describes in depth the decomposition process, but essentially proves that all four matrices can efficiently be learned through gradient decent. Additionally, this process allows long range dependencies as well as discrete representations to be quickly modeled, shown in **??**.

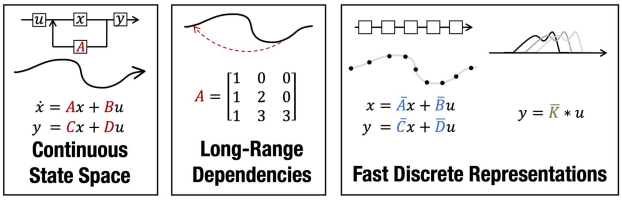
3.2. S4 Decoder

The S4 layer can therefore be used for a wide variety of purposes. Similar to how transformers were repurposed in [4] from a purely natural language application to one that could work with visual tokens, an S4 layer can be applied to a wide variety of inputs. Recently [9] showed that the S4 layer can be used to reduce a very large number of visual tokens (typically in the tens of thousands) from a long down

Figure 2. Sampling Rate of 1

Figure 3. Sampling Rate of 2

Figure 4. Different sampling rates can lead to different ground truth action counts.



to a more manageable number for temporal understanding. In their work, they introduce the S4 Decoder, shown in 6, which reduces both the number of spacial tokens as well as each token's feature dimension while retraining longer term and short term information. We utilize this decoder in our work.

Assume that S4 decoder is given some spaciotemporal feature $X \in \mathbb{R}^{TxHxWxD}$, the S4 decoder block can describe its operation as follows.

$$\begin{aligned} x_{s4} &= S4(LN(x_{in}) \\ x_{mlp} &= MLP(Pooling(x_{s4})) \\ x_{skip} &= Linear(Pooling(x_{in})) \\ x_{out} &= x_{mlp} + x_{skip} \end{aligned} \tag{2}$$

In this formulation, LN refers to a Layer Norm [1] operation Fbeing performed on the sequenced. Then, that normalized sequence is fed to the S4 layer to extract the sequences temporal information. Both the x_{s4} as well as the skip connection x_{skip} are given some max pooling operation. The kernel size of this operation is dependant on the layer. Typically, this kernel is of size $1x2x2$. However, later layers switch to a $1x1x1$ once all spacial tokens have been reduced to 1. Then, both x_{s4} and x_{skip} are fed to a multi-layer perception, and a single linear layer, respectively. This is similar to the typical operation of transformers as described in [16], but aims to also reduce the dimensionality of all features. Then, the temporal information and the skip connection are combined through addition to output the reduced features of the decoder.

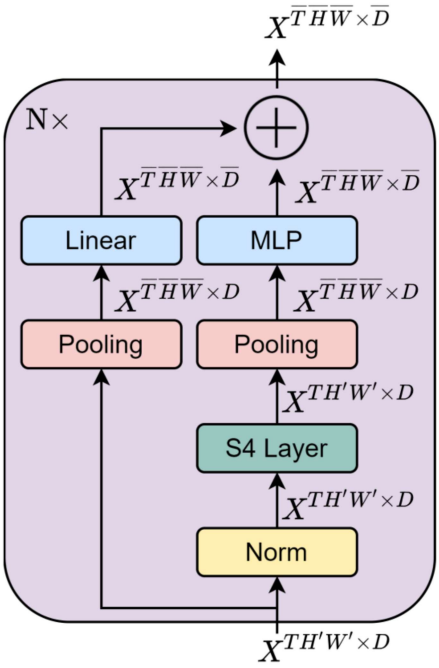


Figure 5. The S4 decoder, as described in [9]

4. Method

4.1. Convolutional Backbone

For the backbone of the model, we use the same backbone as RepNet [5] for most of our experiments, a pretrained Resnet-50 model [8] to generate a set of features for each frame. Resnet-50 takes in frames which have been resize to 112×112, and outputs a set of 2048 features for each frame. The original RepNet model takes the output of the 3rd layer from the Resnet-50 and performs 3D convolution to add some temporal context to the frame features. However, the output of the final Resnet-50 layer can instead

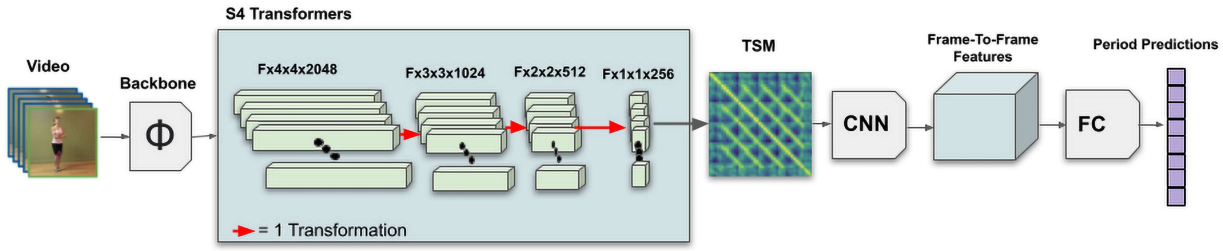


Figure 6. Our approach, using the S4 Decoders to greatly reduce the size of video features while retaining temporal information.

be used to achieve similar results [5], or another model such as the S4 model can be used to add temporal context to the frame features. In this paper, we test multiple different options for the temporal enrichment. If using the output of the Resnet model with no 3D convolution, this will result in 2048 features per frame. If the 3D convolution is used on the 3rd layer of Resnet, this results in 512 features per frame.

4.2. Vision Transformer Backbone

We also test the replacement of the Resnet-50 backbone with the Vision Transformer ViT T/32 [4]. Even the smallest pretrained vision transformer available results in 49 spacial tokens per frame. At the standard frame count of 64, this means that these backbones can often output well over 3000 tokens per video. Many of the videos in the Countix dataset are well over 180 frames, so any sort of self attention between them to understand temporal information is simply not computationally feasible. We perform experiments using the ViT Tiny/32 backbone to show our systems ability to contend with such sequences.

4.3. Temporal Self-Similarity

For the period prediction module, we implement the same model as [5]. This module starts by computing the pair-wise similarities of the frame features, coming from the output of the frame encoder. The pair-wise similarities are calculated using the negative of the L2 distance, followed by a row-wise softmax activation:

$$S_{ij} = -||f_i - f_j||^2 \tag{3}$$

Here, S_{ij} is the similarity between the i-th frame embedding f_i and the j-th frame embedding f_j. These similarities then undergo a row-wise softmax activation, with the final similarities stored in the Temporal Self-Similarity Matrix (TSM):

$$TSM_{ij} = \frac{e^{S_{ij}/\tau}}{\sum_{k=1}^{N} e^{S_{ik}/\tau}} \tag{4}$$

Here, this is the standard softmax function where N is the number of input frame embeddings, and τ denotes the temperature. In this paper, $\tau = 13.544$ and $N = 64$, unless stated otherwise, for the baseline models.

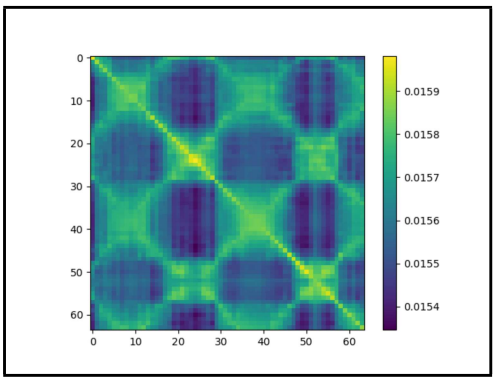


Figure 7. Example TSM of a set pf situps

From [5], this TSM serves to make the frame embeddings interpretable and to provide regularization, as the result is just a 2-dimensional image with a single channel. Once it has been computed, it undergoes a 2-dimensional convolution with a kernel size of 3×3, 32 filters, and a stride of 1, followed by a ReLU activation.

Since there is no straightforward way to get rid of the $O(N^2)$ computational complexity in the TSM, we experiment with using S4 decoders to reduce the number of features for each frame to reduce the overhead in computing the TSM. This also should serve to provide extra temporal context in the TSM.

4.4. Period Prediction

The result of the convolution is a feature map of size $32 \times N \times N$. This is then reshaped into N frame embeddings of size $32 \times N$ which is then projected to 512 features for each frame. Per-frame multi-headed self attention is then calculated on each frames features, using a single transformer layer with an input embedding size of 512, a hidden size of 512, and 4 heads. However, since the computational complexity of the transformer model is $O(N^2)$ with

respect to the number of frames, this can be a major bottleneck when using a lot of frames for counting. Therefore, we also test the replacement of this transformer module with a S4 decoder to improve the runtime.

4.5. Counting

In this paper, since we are only dealing with periodic segments and assuming the same period length for each repetition, a period length is generated for each frame using a Multi-Layer Perceptron (MLP) head with 2 hidden layers of size 512 with ReLU activation, and output size of $\frac{N}{2}$. The final output represents the probability of different period lengths in terms of the number of frames, from 1 to $\frac{N}{2}$.

The final count can then be generated from the period lengths by first calculating an average over each period length l_i for each frame, giving an average period length l_{avg}. Then the count is given by:

$$Count = \frac{l_{avg}}{N} \tag{5}$$

4.6. Evaluation Metrics

There are two main evaluation metrics used for evaluating the performance of repetitive counting models.The first is the mean absolute error (MAE), which measure the difference in count, normalized by the true count, given by Equation (6):

$$MAE = \frac{\sum_{i=1}^{N} |c_i - l_i|/l_i}{N} \tag{6}$$

Here l_i denotes the true count for the i-th video, and c_i denotes the predicted count. The second evaluation metric used is the off-by-one error (OBO). This metric is a misclassification rate which measures how many samples are off by a count of more than one, given by Equation (7)

$$OBO = \frac{\sum_{i=1}^{N} (|c_i - l_i| > 1)}{N} \tag{7}$$

5. Data Augmentation

In this section we go over the data augmentation techniques we used and discuss the parameters used:

5.1. Sampling

Our input to our network starts off by selecting a video from the dataset and sampling it. We do random sampling in a similar manner to the authors [5], that is sampling with a randomly chosen stride of 1 to 4. All of the videos are sampled at random with the consideration that the sampled video clip C should at all times have at least 2 minimum repetitions. This is to make sure the similarity matrix is not sparse.

We found out by sampling randomly it creates a more even distribution of the period length as opposed to the original data being skewed towards larger period length videos. As shown in Figure 8 the data distribution is heavily skewed. This augmentation helps to prevent the model being overfit on the data.

5.2. Horizontal Flip

We randomly augment the whole 64 frames of the input video with horizontal flip to augment the data. The augmentation is done to only a fraction of the complete dataset, with a probability of p. This helps in increasing the volume of the data and also to make sure that the model does not over fit on features specific to the motion but also other motions in the video.

6. Experiments

6.1. Dataset

For all of our experiments, we use the Countix dataset, gathered by the authors of [5]. This dataset is a subset of the Kinetics-400 dataset [11], annotated with sections containing only repetitive motion, as well as a count for each video. In our experiments, we only use these repetitive sections. The resulting dataset has 4332 videos in the training set and 1367 videos in the validation set, which is slightly less than the original Countix due to some videos which were removed from Youtube. The period lengths are assumed to be uniform for the frames in each video, since individual repetitions are not labeled. These period lengths are given by the number of frames divided by the total sampled count, after first ensuring that at least two repetitions have been sampled.

It is clear from Figure 8 that there is a significant imbalance of the count in this dataset. However, this is slightly mitigated with the use of uniform sampling with a random stride on the training set, up to 4x. Also, by forcing the the sampled video to cover at least two repetitions, this creates a wider spread of period lengths due to those videos which get sampled at a 2-4x rate instead of a 1x rate in both the training and validation sets.

6.2. Testing different model configurations

We tested multiple different model configurations, both with and without the use of S4 layers, as well as some models with more than 64 frames. Unless stated otherwise, the models use $N = 64$ frames and output the final period length using a fully-connected network with 1 hidden layer of size 512, and an output size of 32. The following is a description of each of the different architectures tested:

- **Network 1:** This configuration simply takes the output frame encodings from the Resnet-50 backbone and re-

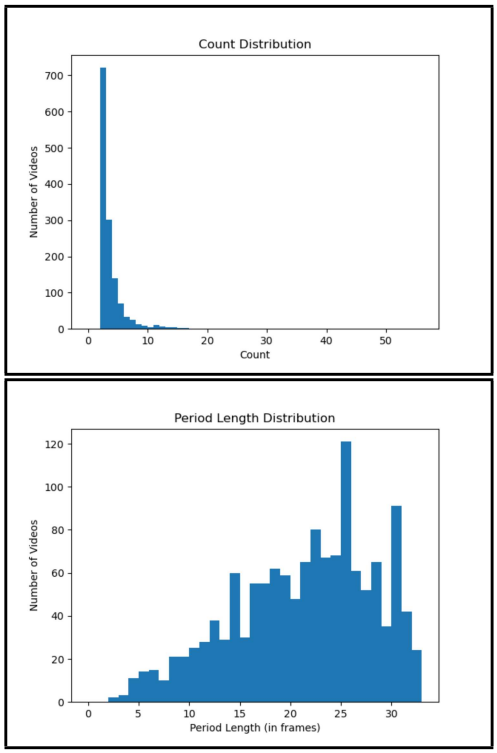


Figure 8. Distribution of count and period length in countix validation set

duces the features from 2048 to 512 with a linear projection. Then the model then adds a one-dimensional positional embedding and forwards the frames through a transformer encoder [16] with 3 layers, 4 heads, and 2048 feedforward size.

- **Network 2:** This configuration is the same as network 1, except it forwards the frame encodings through a S4 decoder instead of a transformer. This S4 decoder has an input and output size of 512, 1 head, and 3 layers.
- **Network 3:** This network uses a S4 decoder with input/output size 512, 1 head, and 3 layers on the output of the Resnet-50 backbone to add temporal context before computing the temporal self-similarity. Then, a 3×3 convolution with 32 channels is applied on the TSM, similar to [5].
- **Network 4:** This network is exactly the same as the original RepNet model, except the 3D convolution is replaced with a S4 decoder as another method of adding temporal context. This S4 decoder has an input size of 1024 from the **third** layer of Resnet-50, an output size of 512, 1 head, and 3 layers.
- **Network 5:** This model is the same as RepNet, except there are two S4 decoders to add temporal context to the frames features from the **final** layer of Resnet-50. The first decoder has an input size of 2048 and output size of 1024. The second decoder has an input size of 1024 and an output size of 512
- **Network 6:** This model uses an S4 decoder on the output of the Resnet-50 backbone to add temporal context and reduce the feature size from 2048 to 512. This model also removes the 2D convolution on the TSM, and instead feeds each row of the TSM through another S4 decoder with input and output size $N = 64$, 4 heads, and 0.2 dropout.
- **Network 6 (4 layer):** This is the same as network 6 above, except the final S4 decoder uses 4 layers instead of 1, to capture a more complex temporal understanding of the video with less features for each frame.
- **Network 7:** This network is the exact same as the above 1 layer network 6, except the the 2D convolution on the TSM is included before being passed on to the final S4 decoder. This 2D convolution has a kernel size of 3×3 and 32 channels, therefore the input size of the final S4 decoder is 32×64 for the 64 frame case.
- **Network 7 (128 frame):** This network is the exact same as the above network 7, except 128 frames are used instead of 64 frames. Therefore the final S4 decoder will have an input size of 32×128 in this case.
- **Network 8:** This model uses an S4 decoder with spatial pooling in place of the 3D convolution in RepNet. This S4 decoder serves the reduce the feature size for each frame before the computation of the temporal self-similarity, which should help to reduce the computational overhead especially for a large number of frames. The S4 decoder consists of 3 layers, the first with input size 2048 and output size 1024, the second with output size 512, and the third with output size 256. The normal transformer encoder is used after the convolution on the TSM, the same as in RepNet.
- **Network 9:** This model removes the initial temporal context altogether, and instead uses 4 layers of S4 decoders on the output of the convolution on the TSM. The first S4 decoder has input size 64×32, and the following three S4 decoder have input/output size 512, all with 4 heads and 0.25 dropout.

For each of these models, we recorded the evaluation metrics on the Countix validation set after exactly 20 epochs of training. Most of these models generally started off generating the same predictions on the validation set because of class imbalance, this causes the validation errors to start off much lower than the training errors and increase overtime. For the majority of these models, overfitting started to

occur well before the 20th epoch, and the validation errors continued to diverge past this point.

In the end, we end up just using network 8, as this produced the best results and allows processing of a much larger number of frames.

6.3. Implementation Details

For each experiment, we use the same hyperparameters as [5]. This corresponds to using the Adam optimizer with a learning rate of $6e-6$ and a batch size of 5 with 64 frames unless stated otherwise. For the coding of our models, we also use the the implementation of the temporal self-similarity matrix provided by the authors of [5].

6.4. Results

Results on COUNTIX		
System	MAE	OBO
RepNet (Reported)	0.36	0.30
RepNet (Tested)	0.46	0.50
Ours (32 Frames)	0.59	0.56
Ours (64 Frames)	0.55	0.48
Ours (128 Frames)	0.53	0.48
Ours (180 Frames)	0.47	0.46

As mentioned, we compare our system both to the results of [5], as well as the results we were able to obtain using their publicly available code. In both MAE and OBO results, we demonstrate a marginal increase in performance as the number of frames our system is fed increases. This makes intuitive sense, as additional frames can either increase confidence in single repetitions due to increase density, or essentially increases the size of the dataset, as our system will learn more about the visual information of each period.

On MAE, our system does not match the performance of our tested results on RepNet's public model, beating our system by several points on each system. However, with our 180 frame model comes very close, only missing by one point. We fair better on the OBO results. Again, we do not come very closer to the results on RepNet's non-public results. However, our 64 frame, 128 frame, and 180 frame models all beat our tested results on their public model. The 180 frame system, in fact, exceeds by four points.

Computing the pairwise temporal self-similarity for each frame embedding can be expensive ($O(N^2)$ for N frames). Therefore, we also test a much larger reduction in the number of features in the S4 transformer. The model is the same as our final S4 Transformer model, but uses 64 frames and a different output size in the final layer of the S4 Transformer.

Results on COUNTIX		
System	MAE	OBO
Ours (256 Features)	0.55	0.48
Ours (64 Features)	0.58	0.52
Ours (1 Features)	1.05	0.70

From the above table, it is clear that the model performs worse when utilizing less features for the input to the TSM module. This is expected since a lot of information is lost when reducing all the way from 512 features to just a single feature. However, the results for using 64 features are still comparable to the results when using 256 features.

6.5. Vision Transformer

Leveraging our systems ability to handle a large number of tokens, we also tested our system using a pretrained Vision Transformer as a backbone. For this, we use ViT tiny with a patch size of 32. We use 6 S4 Decoders to reduce the output feature size from 64x7x7x786 to 64x1x1x256

Results on COUNTIX		
System	MAE	OBO
Ours (ViT backbone)	0.53	0.45

We show a 2 point improvement over our 64 frame model when switching to ViT, which is expected given the supervised recognition performance different versus ResNet50. More importantly however, this shows as proof of concept that this approach toward feature reduction could be applied to more powerful video transformers, whereas previous approaches were required to use 3D convolutional neural networks to satisfy memory requirements.

7. Discussion

7.1. Applications

Predicting occurrence of processes from videos Our proposed algorithm is more suitable for predicting the count and occurrence of any action from videos. It can be specifically focused on biological processes which are repetitive in nature such as heartbeats. [17] utilizes magnifying the changes among the videos frames to identify subtle changes in videos. The output from this can be fed into out network and predict the changes.

Estimating count over larger temporal window In addition to motion occurring in the wild in smaller instances, repetition patterns occur over larger temporal windows which can be repeat as long as days. Our algorithm will be able to predict count in such cases by simply using a larger stride rate.

Sports Analysis Our work can serve as an analysis platform when combined with action localization and action

recognition. It can be utilized in conjunction with [3] to analyze the action instances where a player was fouled and then using our work to analyze localized action instances to count the total number of fouls in a game.

Anomaly Frequency This work can be extended to security analysis where we can count the number of times an anomaly occurs. [15] utilizes real-world surveillance videos which are unprocessed to identify the number of anomalies present in the real world, our work can be used on top of the output to analyze the frequency of anomaly present creating a summary of the events that occurred during a time period.

7.2. Limitations

While our system is able to eliminate some of the bottlenecks in previous models associated with higher frame counts, it still requires each frame be individually encoded in order to learn periodicity. This means that even with the S4 Transformer, extremely large numbers of frames must be split into minibatches for the visual backbone. Future work could investigate how to pool this information earlier in the model sidestep this issue.

7.3. Future Work

Our system embeds temporally rich information into frame encoders while reducing the visual information only to what is necessary to estimate periodicity. Future work could take this approach to its logical conclusion, directly using S4 Transformers to reduce the frame features of a video down to 1, essentially denoting a single output signal. A 1D CNN could then be used to learn the periodicity of the video on this 1D signal, rather than the 2D CNN on the temporal self-similarity matrix.

8. Conclusion

In our work, we have presented a simple yet efficient approach to solve repetitive action counting problem. We utilize the state space transformers to handle the temporal information from the video hence enabling us to process the information in one single pass. This model achieves comparable state of the art results and successfully predicts the count over a complex dataset that has a varied set of objects in motion captured from a diverse set of sensors. We believe given a longer sequence of videos the state space transformers will be able to perform better and handle dense repetition sequence and using only a single pass. The next steps involve using different camera augmentation techniques to handle the camera motion in order to make sure the model does not over fit only on repetitive motion itself. Our work can be extended to counting biological processes, summarizing key events in surveillance and sports analysis.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[2] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. 2

[3] Adrien Deliege, Anthony Cioppa, Silvio Giancola, Meisam J Seikavandi, Jacob V Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B Moeslund, and Marc Van Droogenbroeck. Soccernet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4508–4519, 2021. 8

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 4

[5] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Counting out time: Class agnostic video repetition counting in the wild. *CoRR*, abs/2006.15418, 2020. 1, 2, 3, 4, 5, 6, 7

[6] N Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W Mahoney. Lipschitz recurrent neural networks. *arXiv preprint arXiv:2006.12070*, 2020. 2

[7] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[9] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. *arXiv preprint arXiv:2204.01692*, 2022. 2, 3

[10] Xiaoheng Jiang, Li Zhang, Mingliang Xu, Tianzhu Zhang, Pei Lv, Bing Zhou, Xin Yang, and Yanwei Pang. Attention scaling for crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2

[11] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. 5

[12] Roberto Morelli, Luca Clissa, M. Costa Dalla, Michael Luppi, Lisa Rinaldi, and Antonio Zoccoli. Automatic cell counting in flourescent microscopy using deep learning. *ArXiv*, abs/2103.01141, 2021. 2

[13] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3394–3403, June 2021. 2

[14] Tom FH Runia, Cees GM Snoek, and Arnold WM Smeulders. Real-world repetition estimation by div, grad and curl. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9009–9017, 2018. 2

[15] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488, 2018. 8

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3, 6

[17] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William Freeman. Eulerian video magnification for revealing subtle changes in the world. *ACM transactions on graphics (TOG)*, 31(4):1–8, 2012. 7

[18] Yunhua Zhang, Ling Shao, and Cees G. M. Snoek. Repetitive activity counting by sight and sound. *CoRR*, abs/2103.13096, 2021a. 1, 2