# Lightweight Network Architecture for Real-Time Action Recognition

Alexander Kozlov
alexander.kozlov@intel.com
Intel Corp.
Nizhny Novgorod, Russia

Vadim Andronov*
vadimadr@gmail.com
Intel Corp.
Nizhny Novgorod, Russia

Yana Gritsenko
yana.gritsenko@intel.com
Intel Corp.
Nizhny Novgorod, Russia

## ABSTRACT

In this work we present a new efficient approach to Human Action Recognition called Video Transformer Network (VTN). It leverages the latest advances in Computer Vision and Natural Language Processing and applies them to video understanding. The proposed method allows us to create lightweight CNN models that achieve high accuracy and real-time speed using just an RGB mono camera and general purpose CPU. Furthermore, we explain how to improve accuracy by distilling from multiple models with different modalities into a single model. We conduct a comparison with state-of-the-art methods and show that our approach performs on par with most of them on famous Action Recognition datasets. We benchmark the inference time of the models using the modern inference framework and argue that our approach compares favorably with other methods in terms of speed/accuracy trade-off, running at 56 frames per second (FPS) on CPU. The models and the training code are available[1].

## CCS CONCEPTS

• **Computing methodologies → Activity recognition and understanding**;

## KEYWORDS

Human action recognition, CNN, self-attention, Transformer, OpenVINO

**ACM Reference Format:**
Alexander Kozlov, Vadim Andronov, and Yana Gritsenko. 2020. Lightweight Network Architecture for Real-Time Action Recognition. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341105.3373906

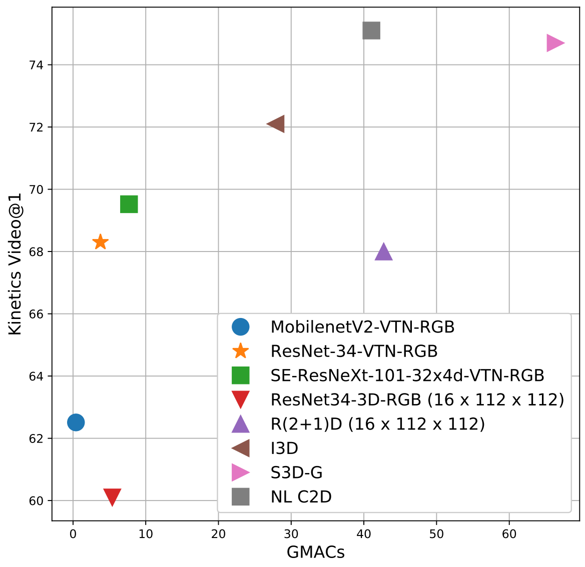

**Figure 1: Accuracy vs complexity trade-off for different methods on Kinetics-400 validation set. First three models are the variants of the proposed VTN method. ResNet-34 3D with a similar number of GMAC[2](accepts smaller resolution inputs) is presented for comparison. We also included several state-of-the-art methods: I3D [9], R(2+1)D [48], S3D-G [54], NL-C2D [53].**

## 1 INTRODUCTION

The latest advances in the Computer Vision domain are definitely related to the development of Deep Learning (DL) methods [24, 43, 46] which show great results on many tasks such as Image Classification [14] and Segmentation [12], Object Detection [16, 33], etc. There is a tendency nowadays to create more and more sophisticated pipelines [7, 23, 58], combining quite complex components which solve the task well but require a massive amount of calculations and power at the same time. On the other hand, since the times of AlexNet [31] and VGG [43] where a vanilla convolution was used as a basic building block, new lightweight primitives have been proposed [10, 27, 28, 57], allowing to reduce the theoretical complexity but retain or even improve the final accuracy. However, video-level tasks, such as Human Action Recognition, which is being discussed in this work, require to consider temporal structure

*At the time of his work in the company.
[1]https://github.com/opencv/openvino_training_extensions/tree/develop/pytorch_toolkit/action_recognition
[2]Billion of multiply-accumulate operations

*SAC '20, March 30-April 3, 2020, Brno, Czech Republic*
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3373906

of input data by aggregating information from multiple frames in order to solve action ambiguities (opening/closing the door). This inevitably incurs extra computational costs during inference of the model. Nevertheless, few studies [8] pay attention to the complexity of the algorithm while maximizing accuracy. Therefore, creating a solution that can achieve high accuracy providing a fast inference speed would be a relevant task, especially in the case of low-power devices used for edge computing (at the edge).

Following this idea, we propose a lightweight architecture for Action Recognition (AR) which can run in real-time on a regular CPU, performing on par with heavy methods, such as 3D CNN [9, 47, 48]. In support of this, we provide a comparison (see Fig. 1 and Section 4.3) of our model with the state-of-the-art methods and verify its accuracy on modern benchmarks, such as Kinetics [29], UCF-101 [44], and HMDB-51 [32].

Shortly, our contributions can be summarized as follows:

- A new lightweight CNN architecture for real-time Action Recognition that achieves results comparable to state-of-the-art methods.
- Comparison of modern approaches to Action Recognition.
- A method for improving the accuracy of an existing model by accommodating information from additional modality without a discernible increase in complexity.

## 2 RELATED WORK

Currently, there are multiple methods that solve the AR problem with certain quality.

One of the examples is the two-stream framework that fuses information from spatial and temporal nets [18, 42]. Spatial net uses RGB frame as input and represents an ordinary classification CNN working on a frame level, whereas temporal net receives multiple stacked optical flow (OF) frames. Calculating OF with traditional algorithms, such as TVL1 [56], requires extra resources, but there are several ways to avoid it. For example, OF can be extracted with additional sub-network [45] or RGB difference [52] can be used as an alternative motion representation.

Another popular group of methods is related to the use of 3D primitives like 3D Convolution, 3D Batch Normalization, 3D Pooling, and others. They generalize original operations introducing an additional dimension $T$, which indicates the sequence of frames. One of the first architectures that leveraged these primitives for the application to AR, is C3D [47]. Another famous 3D CNN, which saturated UCF-101 benchmark [44], is I3D [9]. It benefits from pre-training on a large-scale ImageNet [14] dataset by inflating trained 2D filters into 3D. Although methods based on 3D convolutions allow improving results in terms of accuracy, the computational expenses may achieve dozens of GFLOPs. Another substantial drawback is that at some level of the network only a small number of weights inside the convolutional kernels have a significant impact on the output signal regarding their contribution to the absolute value of activations making utilization of resources ineffective. This problem was mentioned in [48, 54] where authors proposed decomposition techniques and mixed architectures that combine 3D and 2D operations on different levels of the network.

Recurrent neural networks (RNN), LSTMs [26], and GRUs [11] have been regarded as the default starting point for many sequence modeling problems, such as machine translation or language modeling [21]. Many significant results have been achieved in several challenging tasks by means of employing recurrent networks and attention mechanism [4, 41]. Not surprisingly, several approaches to video classification that model sequences with recurrent connections or gated units have been proposed [15, 40, 55]. These models, while showing comparable results on many benchmarks [9], seem to be more suitable for online prediction and thus real-time applications, because feature vector computed for the frame can be reused for predicting classification label for multiple time-windows containing this frame.

Several viable alternative approaches to sequence modeling have been proposed recently. These approaches, for example convolutional [5] or fully-attentional (e.g. Transformer [49]) networks, achieve better results on many tasks while addressing significant shortcomings of RNNs such as sequential computing or gradient vanishing. Later this approach was adopted in [20] to improve the framework of action localization and classification in videos.

We adopt recently proposed Transformer network in our work as a more elaborate way for sequence modeling. This allowed us to attain high accuracy of recognizing human actions, retaining the performance, that is sufficient for real-time applications.

## 3 APPROACH

In this section, we describe a designed approach to AR problem in details as well as discuss some improvements that help to boost the accuracy of our baseline architecture without significantly increasing the complexity.

### 3.1 Architecture overview

Video Transformer Network (see Fig. 2) consists of two parts: the first is the encoder that processes each frame of input sequence independently with 2D CNN in order to get frame embeddings, and the second is the decoder that integrates intra-frame temporal information in a fully-attentional feed-forward fashion, producing the classification label for the given clip. ResNet-34 is used [24] as a baseline architecture for the encoder in most of our experiments. We reuse parameters of all convolutional layers to maximize the benefit of transfer learning from image classification tasks. Global average pooling is then applied to the resulting feature maps to get the frame embeddings of size $d$ (that is equal to 512 in our case), which are then transformed by the decoder, by repeatedly applying multi-head self-attention and convolutional blocks. In multi-head self-attention block, a temporal interrelationship between frames is modeled by informing each frame representation by representation of other frames using the attention mechanism. It consists of several sequential operations. First, vectors of frame representations are mapped to multiple key, value, and query spaces using different learned affine transformations. Each triple of query $Q \in \mathbb{R}^{t \times d_k}$, key $K \in \mathbb{R}^{t \times d_k}$, value $V \in \mathbb{R}^{t \times d_v}$ matrices (where $t$ is the sequence size and $d_k$, $d_v$ are the dimensions of key and value space accordingly) is then transformed to the corresponding head output using the scaled multiplicative attention as following:

$$\text{head}_i = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

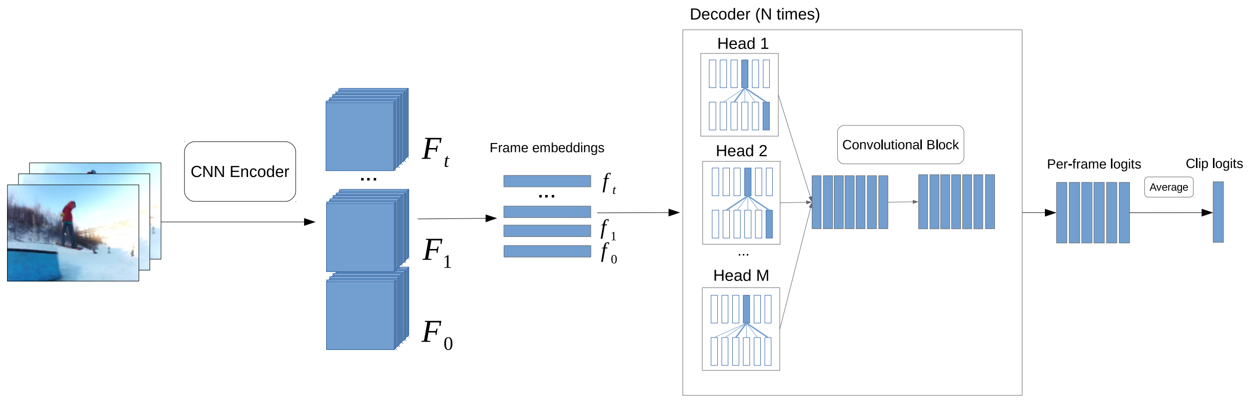

**Figure 2: Overview of the VTN architecture.** ***t*** **input frames are fed to CNN encoder and global pooled to get frame embeddings. Then the decoder block (see in details in Fig. 3) is applied** ***N*** **times. In the end, the clip logits are produced by averaging all frame logits.**

Each head output is then concatenated and passed to the convolutional block that consists of two convolutions with kernel of size 1 (position-wise feedforward) and residual connection. Resulting frame representations are then refined by applying the same procedure multiple times. As we found experimentally, four stacks of such decoder blocks are sufficient for maximizing classification accuracy, and the further increase of the number of blocks did not lead to improvement. In order to produce action confidences for the current clip, a fully-connected layer is applied to all elements of the sequence. Resulting scores are then averaged and normalized with softmax function producing the clip prediction.

## 3.2 Multimodal knowledge distillation

As it was discussed above, the fusion of results of models that receive inputs with different modalities is a common approach to improve the accuracy of Action Recognition algorithm. But in most cases, it leads to a substantial increase in computational complexity due to several reasons. First, it requires to calculate a new modality, which itself may be a hard task, especially in case of the optical flow where commonly used algorithms perform costly iterative energy minimization. Second, since the same architecture is used to do prediction using the second modality, the complexity of the method is doubled. Therefore, both issues make applying of multimodal solutions difficult in real-world applications.

On the other hand, using the RGB difference in place of the optical flow results in almost the same performance [52], which has been verified by our experiments. At the same time, it requires much lower computational resources that makes using this modality more suitable in conjunction with a still RGB data.

Knowledge distillation [25] is the procedure that designated to help optimization of the student network by providing extra supervision from a larger model or an ensemble of models (teacher). There are successful applications of this technique for reducing the complexity of a larger teacher network [37] or integrating the performance of an ensemble of models into a single student [6, 25]. However, we hypothesize whether it is possible to transfer knowledge from multiple models working on different modalities (two-stream teacher) to a single student. In order to better understand this, we ran several experiments where knowledge from two ResNet-34 based VTN models working with RGB and RGB difference is distilled to the single RGB model and to the model which receives stacked RGB and RGB difference inputs. We also tried to train a model that operates on stacked input without extra supervision from knowledge distillation. Results are shortly summarized in Table 1. The model working on stacked inputs outperforms the single modality model when trained with knowledge distillation. We suppose that the main reason for that is that motion representation, learned by RGB-difference subnetwork in the two-stream teacher, are not discovered by RGB-only model, yet they significantly contribute to model performance. Note that this technique does not allow matching the performance of the two-stream model. However, it significantly reduces the complexity compared with the original two-stream solution.

**Table 1: Results of knowledge distillation (KD) from two-stream (fusion of two models) ResNet-34-VTN teacher on Mini-Kinetics dataset. The single model that works with stacked modalities improves its accuracy when trained as a student in knowledge distillation setup. However, RGB-only model does not benefit from KD.**

| Model | Video@1 | GMAC |
|---|---|---|
| Fused RGB + RGB-diff (teacher) | 78.2 | 7.51 |
| RGB | 75.2 | 3.77 |
| RGB with KD | 75.2 | 3.77 |
| Stacked RGB + RGB-diff | 75.2 | 3.88 |
| Stacked RGB + RGB-diff with KD | 76.0 | 3.88 |

## 4 EXPERIMENTS

In this section we present a study of the proposed method. Kinetics-400 is considered as the primary benchmark. However, the smaller Mini-Kinetics subset that was introduced in [54] is also used for

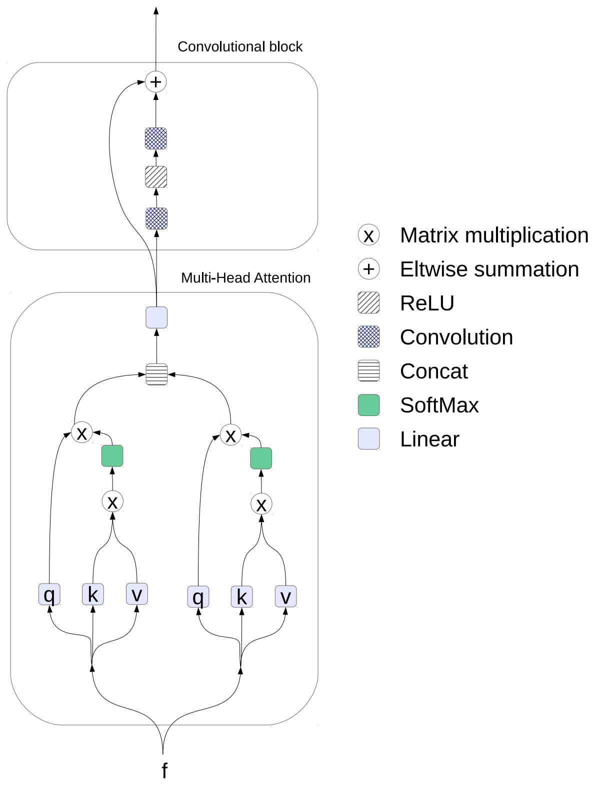

**Figure 3: The detailed overview of decoder block used in VTN. We use $M = 2$ self-attention heads on the scheme for simplicity. Each head independently transforms input sequence embeddings to its query (q), key (k), value (v) triplet (that are rows of Q, K, and V matrices accordingly) using three trainable linear transformations and applies the self-attention operation. In order to produce output sequence, resulting vectors are concatenated and passed to the block of two convolutions with the kernel of size 1 and residual connection around those convolutions.**

faster experimentation. We also evaluated our models on UCF-101 and HMDB-51 and evaluated the inference speed on CPU.

## 4.1 Implementation details

We train and validate our models on 16-frame input sequences that are formed by sampling every second frame from the original video, therefore the total temporal receptive field of our model equals to 32 frames. We tried longer sequences by adding or skipping more frames, but this only resulted in an increased clip accuracy, not the video. In order to calculate video classification accuracy (Video@1), we extracted all non-overlapping 32 frame segments and averaged prediction on these segments.

Frames are scaled in a way, that the shorter side becomes equal to 256. We randomly crop $224 \times 224$ with four different scales during training, as described in [51], and use central $224 \times 224$ crop during the test time. Adam optimizer [30] with the momentum of 0.9 and weight decay of 0.0001 is used. Training is started with the learning rate of $10^{-4}$, which is decayed by a factor of 10 when validation loss reaches a plateau. Models are trained until validation loss stops decreasing, which is usually happened within 50 epochs.

## 4.2 Model hyperparameters

We varied the structure of our decoder block in order to come up with one that maximizes performance on Mini-Kinetics dataset and believed that the same parameter settings would maximize efficiency on other datasets.

First of all, we evaluated how the number of stacked decoder blocks affects accuracy. We trained models with 1,3,4,5 and 6 blocks, and determined that 4 blocks result in the maximal accuracy and the higher number of blocks does not further boost the metric. We also experimented with sharing parameters between blocks by applying one block recurrently, as suggested in [13], but it did not lead to performance improvement. We varied the number of heads in multi-head self-attention, and dimension of query, key $d_k$, and value $d_v$ space, $M = 8$ heads with $d_k = d_v = \frac{d}{M}$ gave the best results. We also tried to add trainable linear transformation after concatenation of heads and to use layer normalization in different locations, but these changes did not affect the accuracy.

## 4.3 Comparison with other methods

In order to better understand capabilities of the proposed approach, we compare it with methods described in Section 2. For a fair comparison, we take ResNet-34 architecture and extend it to the case of 3D networks and two-stream methods in the way described below.

The first model we compare with is ResNet-34 3D which is described in [22]. It repeats a common ResNet architecture, but instead of 2D Convolutions and Pooling layers, it utilizes their 3D analogs. A global Average Pooling operation over three dimensions is applied at the end of the network in order to get a representation vector, which is fed to a fully-connected layer producing the CNN output. Vanilla ResNet-34 pre-trained on ImageNet is used to initialize its 3D analog where convolutional kernels are repeated over temporal dimension $T$, as proposed in [9].

The next approach that we consider is a two-stream model that is represented by a fusion of two ResNet-34 CNNs trained on RGB and OF inputs. The OF model is almost the original ResNet-34, but its first convolutional layer receives 32-channels input, formed by $X$ and $Y$ components of pre-calculated optical flow for 16 sequential frames. To initialize this layer we average the first convolutional kernel of the RGB model pre-trained on ImageNet over the channel dimension and repeat it 32 times.

We also tried a two-stream model where two fused CNNs were trained on RGB and RGB difference inputs since the calculation of the latter is much cheaper than the optical flow. In this case, the motion model receives 48-channels input of RGB differences from 16 consecutive frames.

The last model examined in our comparison is the ResNet-34 followed by three stacked LSTM cells operating on independent frame embeddings. As before, we use the ImageNet pre-trained model for initialization, but learn LSTM parameters from scratch. We found

**Table 2: Comparison of different approaches to Action Recognition on Mini-Kinetics dataset with further finetuning on UCF-101 split 1 (Accuracy Video@1). All models are based on the ResNet-34, with the input resolution of 224x224 and 16-frame inputs. Inference time was measured on Intel Core$^{TM}$ i7-8700 CPU @ 2.90GHz and expressed in Frames Per Seconds.**

| Model | Mini-Kinetics | UCF-101 | GMAC | FPS | Million Parameters |
|---|---|---|---|---|---|
| 3D CNN | 72.9 | 86.4 | 50.2 | 5 | 63.5M |
| Fused RGB and OF | 74.3 | 89.8 | 8.5[3] | 32 | 42.8M |
| Fused RGB and RGB-diff | 73.7 | 88.3 | 9.1 | 30 | 42.9M |
| Stacked LSTMs | 72.0 | 86.6 | 3.7 | 55 | 27.6M |
| VTN (ours) | 75.2 | 89.0 | 3.8 | 56 | 29.0M |

**Table 3: Comparison with the state-of-the-art on Kinetics-400 dataset.**

| Method | Video@1 |
|---|---|
| BNInception+TSN-RGB [52] | 69.1[4] |
| I3D-RGB [9] | 72.1 |
| I3D-TwoStream [9] | 75.7 |
| S3D-G [54] | 74.7 |
| R(2+1)D-TwoStream [48] | 75.4 |
| R(2+1)D-RGB [48] | 74.3 |
| NL-I3D-ResNet-101-RGB [53] | 77.7 |
| MobileNetV2-VTN-RGB | 62.5 |
| ResNet-34-VTN-RGB | 68.3 |
| ResNet-34-VTN-RGB+RGBDiff | 71.0 |
| SE-ResNeXt-101-VTN-RGB | 69.5 |
| SE-ResNeXt-101-VTN-RGB+RGDiff | 73.5 |

this model quite simple but representative at the same time. We also tried to apply a visual attention mechanism, as suggested in [40], but it did not improve the performance.

The comparison of the described models and our proposed method is shown in Table 2. For the sake of convenience, we also provide a theoretical complexity and inference time for all models. The input resolution is set to 224x224, and the sequence size is 16 frames for all models. The models were trained with Adam optimizer until validation loss reaches the plateau. The obtained results show that our VTN model outperforms others on Mini-Kinetics dataset and works on par with the two-stream method. We find this fact surprising because we believe that 3D Convolutional model should perform better because it consists of operations that can learn temporal dependencies at every layer and has a higher capacity regarding the number of parameters.

Another interesting result is that the two-stream RGB-difference model shows the performance that is close to the OF-based model while saving a large number of calculations. These findings correspond to the results of [22, 36]. Nevertheless, our VTN approach is attractive in terms of speed/accuracy trade-off.

## 4.4 Comparison with state-of-the-art

To compare with other state-of-the-art models we assessed our approach on Kinetics-400 dataset. In addition to the baseline ResNet-34-VTN, we used a larger model employing SE-ResNeXt-101 (32x4d) architecture for the encoder, which is, however, still very cheap in terms of a number of multiply-accumulates in comparison with 3D CNNs. Another interesting question is the potential of the proposed method in optimizing a model for mobile devices and what associated drop in accuracy it would incur. To tackle this question we tested our approach with the lightweight MobileNetV2 [39] encoder.

Since fusion of prediction from streams with different modalities (e.g. RGB and optical flow or RGB and RGB difference) allowed improving results in many published works, we experimented with enhancing the results of our RGB model by combining it with the analogous RGB difference model. We subtracted normalized adjacent frames and trained the ResNet34-VTN model on this data. This allowed us to improve the results of the ResNet34-VTN model by a margin of 2.4%.

The results for the Kinetics-400 validation set are presented in Table 3. The breakthrough I3D model [9] outperforms ResNet-34 VTN and SE-ResNeXt-101 (32x4d) VTN only by a small margin of 3.5% and 2.1% accordingly, thus our method still shows competitive results while being computationally significantly cheaper for online prediction scenarios.

We also provide results on the popular UCF-101 and HMDB-51 datasets. We fine-tuned models trained on Kinetics-400 for 20 epochs with smaller learning rate of $10^{-5}$. Mean video accuracies over three validation splits are presented in Table 4.

Computational complexity versus accuracy on Kinetics-400 for some state-of-the-art methods and various variants of VTN is shown in Fig. 1. Since we primarily focus on the online prediction scenario (i.e. when the classification label is required for every subsequent frame) we consider the number of operations needed to execute the encoder on one frame as well as operations for the whole decoder. On the other hand, 3D convolutional models extract features from adjacent frames and require to execute the entire network for each new frame. Thus our method is more attractive in terms of accuracy/complexity for real-time applications.

## 4.5 Inference speed

Since theoretically faster models do not necessarily correspond to higher inference speed [34, 35, 38], we also evaluate the actual inference time to prove the feasibility of the proposed method for real-time applications. Currently, there are several frameworks

[3]Optical flow calculation is not included in the complexity estimation.
[4]Author's implementation (https://github.com/yjxiong/tsn-pytorch) uses 10-crop TTA during testing.

**Table 4: Comparison with other methods (Accuracy Video@1) on UCF-101 and HMDB-51 (average metric over all splits). Methods of the first set of rows do not use Kinetics pre-training.**

| Method | UCF-101 | HMDB-51 |
|---|---|---|
| IDT [50] | 86.4 | 61.7 |
| C3D [47] | 85.2 | - |
| Two-Stream [42] | 88.0 | 59.4 |
| Two-Stream Fusion + IDT [19] | 93.5 | 69.2 |
| BNInception+TSN-RGB [52] | 91.1 | - |
| P3D [52] | 88.6 | - |
| ST-ResNet + IDT [17] | 94.6 | 70.3 |
| I3D-RGB [9] | 95.6 | 74.8 |
| I3D-TwoStream [9] | 98.0 | 80.7 |
| S3D-G [54] | 96.8 | 75.9 |
| R(2+1)D-TwoStream [48] | 97.3 | 78.7 |
| ResNet-34-VTN-RGB | 90.8 | 63.5 |
| SE-ResNeXt-101-VTN-RGB | 92.2 | 67.2 |
| ResNet-34-VTN-RGB+RGBDiff | 95.0 | 71.3 |
| SE-ResNeXt-101-VTN-RGB+ RGBDiff | 95.0 | 71.6 |

**Table 5: Inference time of various Video Transformer Networks with OpenVINO on Intel Core$^{TM}$ i7-8700 CPU @ 2.90GHz.**

| Model | FPS | GMAC |
|---|---|---|
| ResNet-34-VTN-RGB | 56 | 3.77 |
| Stacked RGB+RGBDiff ResNet-34 VTN | 51 | 4.2 |
| ResNet-50-VTN-RGB | 49 | 4.25 |
| MobileNetV2-VTN-RGB | 177 | 0.4 |

available, such as Nvidia Tensor RT [1] or Intel® OpenVINO$^{TM}$ Toolkit [3], which can highly optimize DL model for particular hardware. Since we primarily focus on models suitable for edge computing, we chose OpenVINO and its DL Deployment Toolkit as the inference engine for our solution. OpenVINO can import models from many DL frameworks as well as ONNX [2] representation which we use to convert models from PyTorch framework which is used in all our experiments.

Table 5 shows the inference time on CPU of several models that employ the proposed approach. Faster than real-time speed is achieved for all models, making this method promising for edge computing.

## 5 CONCLUSIONS

In this work, we have proposed a new Video Transformer Network architecture for real-time Action Recognition. We have shown that adopting methods from Natural Language Processing along with using an appropriate CNN for Image Classification helps to achieve accuracy on-par with state-of-the-art methods. Moreover, it has been demonstrated that the proposed approach favorably compares with other approaches, such as 3D Convolution-based models or two-stream methods. Specifically, it allows utilizing computational resources more effectively by embedding each input frame to lower-dimensional high-level feature vector and then making a conclusion about the action operating only on embedding vectors by means of self-attention. This method allows achieving real-time inference on a general-purpose CPU, providing capabilities for using AR algorithms at the edge. Our research also demonstrates that the self-attention mechanism is quite universal and can be applied to many tasks, such as Natural Language Processing, Speech Recognition or Computer Vision.

## REFERENCES

[1] [n. d.]. NVIDIA TensorRT Programmable Inference Accelerator. https://developer.nvidia.com/tensorrt.

[2] [n. d.]. ONNX. https://onnx.ai/.

[3] [n. d.]. OpenVINO Toolkit. https://software.intel.com/en-us/openvino-toolkit.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[5] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[6] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 535–541.

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2016. Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1611.08050* (2016).

[8] Joao Carreira, Viorica Patraucean, Laurent Mazare, Andrew Zisserman, and Simon Osindero. 2018. Massively Parallel Video Networks. In *The European Conference on Computer Vision (ECCV)*.

[9] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 4724–4733.

[10] F. Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *CVPR*. 1251–1258.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3213–3223.

[13] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819* (2018).

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

[15] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2625–2634.

[16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. 2010. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.

[17] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. 2016. Spatiotemporal residual networks for video action recognition. In *Advances in neural information processing systems*. 3468–3476.

[18] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1933–1941.

[19] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1933–1941.

[20] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. 2019. Video action transformer network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 244–253.

[21] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[22] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. 2018. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA*. 18–22.

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2980–2988.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:arXiv:1503.02531

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[27] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861. arXiv:arXiv:1704.04861

[28] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360* (2016).

[29] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).

[30] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[32] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. 2011. HMDB: a large video database for human motion recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2556–2563.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*. 740–755.

[34] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2755–2763.

[35] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164* (2018).

[36] AJ Piergiovanni and Michael S. Ryoo. 2018. Representation Flow for Action Recognition. arXiv:arXiv:1810.01455

[37] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* (2014).

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *arXiv preprint arXiv:1801.04381* (2018).

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.

[40] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. 2016. Action recognition using visual attention. (2016).

[41] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[42] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*. 568–576.

[43] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:arXiv:1409.1556

[44] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[45] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. 2018. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)(June 2018)*.

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[47] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 4489–4497.

[48] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6450–6459.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[50] Heng Wang and Cordelia Schmid. 2013. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*. 3551–3558.

[51] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. 2015. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159* (2015).

[52] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*. Springer, 20–36.

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2017. Non-local neural networks. *arXiv preprint arXiv:1711.07971* 10 (2017).

[54] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 305–321.

[55] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4694–4702.

[56] Christopher Zach, Thomas Pock, and Horst Bischof. 2007. A duality based approach for realtime TV-L 1 optical flow. In *Joint Pattern Recognition Symposium*. Springer, 214–223.

[57] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2017. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. arXiv:arXiv:1707.01083

[58] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2881–2890.