Video Action Transformer Network

Rohit Girdhar[1*] João Carreira[2] Carl Doersch[2] Andrew Zisserman[2,3]
[1]Carnegie Mellon University [2]DeepMind [3]University of Oxford
http://rohitgirdhar.github.io/ActionTransformer

Abstract

We introduce the Action Transformer model for recognizing and localizing human actions in video clips. We repurpose a Transformer-style architecture to aggregate features from the spatiotemporal context around the person whose actions we are trying to classify. We show that by using high-resolution, person-specific, class-agnostic queries, the model spontaneously learns to track individual people and to pick up on semantic context from the actions of others. Additionally its attention mechanism learns to emphasize hands and faces, which are often crucial to discriminate an action – all without explicit supervision other than boxes and class labels. We train and test our Action Transformer network on the Atomic Visual Actions (AVA) dataset, outperforming the state-of-the-art by a significant margin using only raw RGB frames as input.

1. Introduction

In this paper, our objective is to both localize and recognize human actions in video clips. One reason that human actions remain so difficult to recognize is that inferring a person's actions often requires understanding the people and objects around them. For instance, recognizing whether a person is 'listening to someone' is predicated on the existence of another person in the scene saying something. Similarly, recognizing whether a person is 'pointing to an object', or 'holding an object', or 'shaking hands'; all require reasoning jointly about the person and the animate and inanimate elements of their surroundings. Note that this is not limited to the context at a given point in time: recognizing the action of 'watching a person', after the watched person has walked out of frame, requires reasoning over time to understand that our person of interest is actually looking at someone and not just staring into the distance.

Thus we seek a model that can determine and utilize such contextual information (other people, other objects) when determining the action of a person of interest. The Transformer architecture from Vaswani *et al.* [43] is one suitable model for this, since it explicitly builds contextual support for its representations using self-attention. This architecture has been hugely successful for sequence modelling tasks compared to traditional recurrent models. The question, however, is: how does one build a similar model for human action recognition?

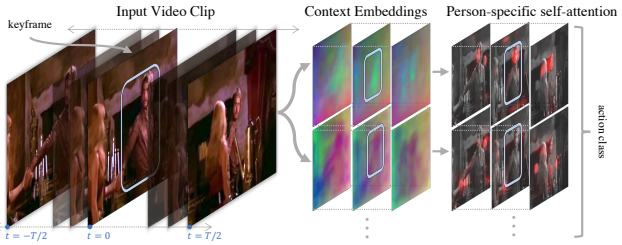

Figure 1: **Action Transformer in action.** Our proposed multi-head/layer Action Transformer architecture learns to attend to relevant regions of the person of interest and their context (other people, objects) to recognize the actions they are doing. Each head computes a clip embedding, which is used to focus on different parts like the face, hands and the other people to recognize that the person of interest is 'holding hands' and 'watching a person'.

Our answer is a new video action recognition network, the **Action Transformer**, that uses a modified Transformer architecture as a 'head' to classify the action of a person of interest. It brings together two other ideas: (i) a spatio-temporal I3D model that has been successful in previous approaches for action recognition in video [7] – this provides the base features; and (ii) a region proposal network (RPN) [33] – this provides a sampling mechanism for localizing people performing actions. Together the I3D features and RPN generate the query that is the input for the Transformer head that aggregates contextual information from other people and objects in the surrounding video. We describe this architecture in detail in section 3. We show in section 4 that the trained network is able to learn both to track individual people and to contextualize their actions in terms of the actions of other people in the video. In addition, the transformer attends to hand and face regions, which is reassuring because we know they have some of the most relevant features when discriminating an action. All of this is

*Work done during an internship at DeepMind

obtained without explicit supervision, but is instead learned during action classification.

We train and test our model on the Atomic Visual Actions (AVA) [15] dataset. This is an interesting and suitable testbed for this kind of contextual reasoning. It requires detecting multiple people in videos semi-densely in time, and recognizing multiple basic actions. Many of these actions often cannot be determined from the person bounding box alone, but instead require inferring relations to other people and objects. Unlike previous works [3], our model learns to do so without needing explicit object detections. We set a new record on the AVA dataset, improving performance from 17.4% [41] to 25.0% mAP. The network only uses raw RGB frames, yet it outperforms all previous work, including large ensembles that use additional optical flow and sound inputs. At the time of submission, ours was the top performing approach on the ActivityNet leaderboard [6].

However, we note that at 25% mAP, this problem, or even this dataset, is far from solved. Hence, we rigorously analyze the failure cases of our model in Section 5. We describe some common failure modes and analyze the performance broken down by semantic and spatial labels. Interestingly, we find many classes with relatively large training sets are still hard to recognize. We investigate such tail cases to flag potential avenues for future work.

2. Related Work

Video Understanding: Video activity recognition has evolved rapidly in recent years. Datasets have become progressively larger and harder: from actors performing simple actions [13, 35], to short sports and movie clips [26, 40], finally to diverse YouTube videos [1, 25]. Models have followed suit, from hand-crafted features [27, 44] to deep end-to-end trainable models [7, 24, 45, 46, 48]. However, much of this work has focused on trimmed action recognition, i.e., classifying a short clip into action classes. While useful, this is a rather limited view of action understanding, as most videos involve multiple people performing multiple different actions at any given time. Some recent work has looked at such fine-grained video understanding [8, 19, 23, 39], but has largely been limited to small datasets like UCF-24 [39, 40] or JHMDB [21]. Another thread of work has focused on temporal action detection [37, 38, 49]; however, it does not tackle the tasks of person detection or person-action attribution.

AVA dataset and methods: The recently introduced AVA [15] dataset has attempted to remedy this by introducing 15-minute long clips labeled with all people and their actions at one second intervals. Although fairly new, various models [15, 22, 41, 51] have already been proposed for this task. Most models have attempted to extend object detection frameworks [16, 20, 33] to operate on videos [10, 19, 23]. Perhaps the closest to our approach is the concurrent work on person-centric relation networks [41], which learns to relate person features with the video clip akin to relation networks [34]. In contrast, we propose to use person detections as queries to seek out regions to aggregate in order to recognize their actions, and outperform [41] and other prior works by a large margin.

Attention for action recognition: There has been a large body of work on incorporating attention in neural networks, primarily focused on language related tasks [43, 50]. Attention for videos has been pursued in various forms, including gating or second order pooling [11, 29, 30, 48], guided by human pose or other primitives [4, 5, 11, 12], region-graph representations [18, 47], recurrent models [36] and self-attention [46]. Our model can be thought of as a form of self-attention complementary to these approaches. Instead of comparing all pairs of pixels, it reduces one side of the comparison to human regions, and can be applied on top of a variety of base architectures, including the previously mentioned attentional architectures like [46].

3. Action Transformer Network

In this section we describe the overall design of our new Action Transformer model. It is designed to detect all persons, and classify all the actions they are doing, at a given time point ('keyframe'). It ingests a short video clip centered on the keyframe, and generates a set of human bounding boxes for all the people in the central frame, with each box labelled with all the predicted actions for the person.

The model consists of a distinct base and head networks, similar to the Faster R-CNN object detection framework [33]. The base, which we also refer to as trunk, uses a 3D convolutional architecture to generate features and region proposals (RP) for the people present. The head then uses the features associated with each proposal to predict actions and regresses a tighter bounding box. Note that, importantly, both the RPN and bounding box regression are action agnostic. In more detail, the head uses the feature map generated by the trunk, along with the RPN proposals, to generate a feature representation corresponding to each RP using RoIPool [20] operations. This feature is then used classify the box into C action classes or background (total $C+1$), and regress to a 4D vector of offsets to convert the RPN proposal into a tight bounding box around the person. The base is described in Section 3.1, and the transformer head in Section 3.2. We also describe an alternative I3D Head in Section 3.3, which is a more direct analogue of the Faster-RCNN head. It is used in the ablation study. Implementation details are given in Section 3.4.

3.1. Base network architecture

We start by extracting a T-frame (typically 64) clip from the original video, encoding about 3 seconds of context around a given keyframe. We encode this input using a

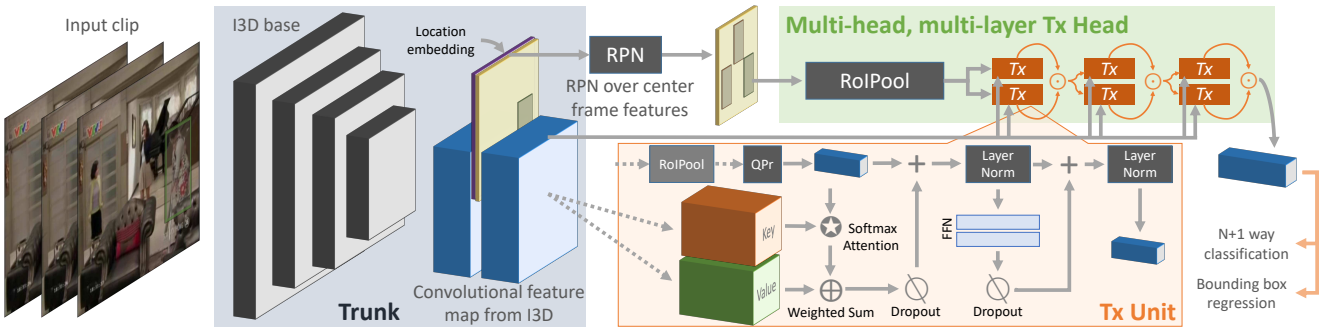

Figure 2: **Base Network Architecture.** Our model takes a clip as input and generates a spatio-temporal feature representation using a trunk network, typically the initial layers of I3D. The center frame of the feature map is passed through an RPN to generate bounding box proposals, and the feature map (padded with location embedding) and each proposal are passed through 'head' networks to obtain a feature for the proposal. This feature is then used to regress a tight bounding box and classify into action classes. The head network consists of a stack of Action Transformer (Tx) units, which generates the features to be classified. We also visualize the Tx unit zoomed in, as described in Section 3.2. QPr and FFN refer to Query Preprocessor and a Feed-forward Network respectively, also explained Section 3.2.

set of convolutional layers, and refer to this network as the trunk. In practice, we use the initial layers of an I3D network pre-trained on Kinetics-400 [7]. We extract the feature map from the `Mixed_4f` layer, by which the $T \times H \times W$ input is downsampled to $T' \times H' \times W' = \frac{T}{4} \times \frac{H}{16} \times \frac{W}{16}$. We slice out the temporally-central frame from this feature map and pass it through a region proposal network (RPN) [33]. The RPN generates multiple potential person bounding boxes along with objectness scores. We then select R boxes (we use $R = 300$) with the highest objectness scores to be further regressed into a tight bounding box and classified into the action classes using a 'head' network, as we describe next. The trunk and RPN portions of Figure 2 illustrate the network described so far.

3.2. Action Transformer Head

As outlined in the Introduction, our head architecture is inspired and re-purposed from the Transformer architecture [43]. It uses the person box from the RPN as a 'query' to locate regions to attend to, and aggregates the information over the clip to classify their actions. We first briefly review the Transformer architecture, and then describe our Action Transformer head framework.

Transformer: This architecture was proposed in [43] for seq2seq tasks like language translation, to replace traditional recurrent models. The main idea of the original architecture is to compute *self-attention* by comparing a feature to all other features in the sequence. This is carried out efficiently by not using the original features directly. Instead, features are first mapped to a query (Q) and memory (key and value, K & V) embedding using linear projections, where typically the query and keys are lower dimensional. The output for the query is computed as an attention weighted sum of values V, with the attention weights obtained from the product of the query Q with keys K. In practice, the query here was the word being translated, and the keys and values are linear projections of the input sequence and the output sequence generated so far. A location embedding is also added to these representations in order to incorporate positional information which is lost in this non-convolutional setup. We refer the readers to [43] and [31] for a more detailed description of the original architecture.

Action Transformer: We now describe our re-purposed Transformer architecture for the task of video understanding. Our transformer unit takes as input the video feature representation and the box proposal from RPN and maps it into query and memory features. Our problem setup has a natural choice for the query (Q), key (K) and value (V) tensors: the person being classified is the query, and the clip around the person is the memory, projected into key and values. The unit then processes the query and memory to output an updated query vector. The intuition is that the self-attention will add context from other people and objects in the clip to the query vector, to aid with the subsequent classification. This unit can be stacked in multiple heads and layers similar to the original architecture [43], by concatenating the output from the multiple heads at a given layer, and using the concatenated feature as the next query. This updated query is then used to again attend to context features in the following layer. We show this high-level setup and how it fits into our base network highlighted in green in Figure 2, with each Action Transformer unit denoted as 'Tx'. We now explain this unit in detail.

The key and value features are simply computed as linear projections of the original feature map from the trunk, hence each is of shape $T' \times H' \times W' \times D$. In practice, we extract the RoIPool-ed feature for the person box from the center clip, and pass it through a query preprocessor (QPr) and a linear layer to get the query feature of size $1 \times 1 \times D$. The QPr could directly average the RoIpool feature across

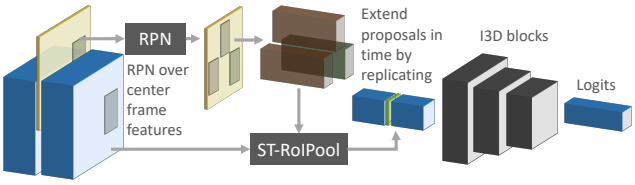

Figure 3: **I3D Head.** Optionally, we can replace the Action Transformer head with a simpler head that applies the last few I3D blocks to the region features, as described in Section 3.3.

space, but would lose all spatial layout of the person. Instead, we first reduce the dimensionality by a 1×1 convolution, and then concatenate the cells of the resulting 7×7 feature map into a vector. Finally, we reduce the dimensionality of this feature map using a linear layer to 128D (the same as the query and key feature maps). We refer to this procedure as **HighRes** query preprocessing. We compare this to a QPr that simply averages the feature spatially, or **LowRes** preprocessing, in Section 4.3.

The remaining architecture essentially follows the Transformer. We use feature $Q^{(r)}$ corresponding to the RPN proposal r, for dot-product attention over the K features, normalized by $\sqrt{D}$ (same as [43]), and use the result for weighted averaging ($A^{(r)}$) of V features. This operation can be succinctly represented as

$$a^{(r)}_{xyt} = \frac{Q^{(r)} K^T_{xyt}}{\sqrt{D}}; A^{(r)} = \sum_{x,y,t} \left[\text{Softmax}\left(a^{(r)}\right)\right]_{xyt} V_{xyt}$$

We apply a dropout to $A^{(r)}$ and add it to the original query feature. The resulting query is passed through a residual branch consisting of a LayerNorm [2] operation, followed by a Feed Forward Network (FFN) implemented as a 2-layer MLP and dropout. The final feature is passed through one more LayerNorm to get the updated query (Q''). Figure 2 (Tx unit) illustrates the unit architecture described above, and can be represented as

$$Q^{(r)'} = \text{LayerNorm}\left(Q^{(r)} + \text{Dropout}\left(A^{(r)}\right)\right)$$
$$Q^{(r)''} = \text{LayerNorm}\left(Q^{(r)'} + \text{Dropout}\left(\text{FFN}\left(Q^{(r)'}\right)\right)\right)$$

3.3. I3D Head

To measure the importance of the context gathered by our Action Transformer head, we also built a simpler head architecture that does not extract context. For this, we extract a feature representation corresponding to the RPN proposal from the feature map using a Spatio-Temporal RoIPool (ST-RoIPool) operation. It's implemented by first stretching the RP in time by replicating the box to form a tube. Then, we extract a feature representation from feature map at each time point using the corresponding box from the tube using the standard RoIPool operation [20], similar to previous works [10]. The resulting features across time are stacked to get a spatio-temporal feature map corresponding to the tube. It is then passed through the layers of the I3D network that were dropped from the trunk (i.e., `Mixed_5a` to `Mixed_5c`). The resulting feature map is then passed through linear layers for classification and bounding box regression. Figure 3 illustrates this architecture.

3.4. Implementation Details

We develop our models in Tensorflow, on top of the TF object detection API [20]. We use input spatial resolution of 400×400px and temporal resolution (T) of 64. The RoIPool used for both the I3D and Action Transformer head generates a 14×14 output, followed by a max pool to get a 7×7 feature map. Hence, the I3D head input ends up being $16 \times 7 \times 7$ in size, while for Action Transformer we use the 7×7 feature as query and the full $16 \times 25 \times 25$ trunk feature as the context. As also observed in prior work [31, 43], adding a location embedding in such architectures is very beneficial. It allows our model to encode spatiotemporal proximity in addition to visual similarity, a property lost when moving away from traditional convolutional or memory-based (eg. LSTM) architectures. For each cell in the trunk feature map, we add explicit location information by constructing vectors: $[h, w]$ and $[t]$ denoting the spatial and temporal location of that feature, computed with respect to the size and relative to the center of the feature map. We pass each through a 2-layer MLP, and concatenate the outputs. We then attach the resulting vector to the trunk feature map along channel dimension. Since K, V are projections the trunk feature map, and Q is extracted from that feature via RoIPool, all of these will implicitly contain the location embedding. Finally, for classification loss, we use separate logistic losses for each action class, implemented using sigmoid cross-entropy, since multiple actions can be active for a given person. For regression, we use the standard smooth L1 loss. For the Action Transformer heads, we use feature dimensionality of $D = 128$ and dropout of 0.3. We use a 2-head, 3-layer setup for the Action Transformer units by default, though we ablate other choices in the arXiv version [9].

3.5. Training Details

Pre-training: We initialize most of our models by pre-training the I3D layers separately on the large, well-labeled action classification dataset Kinetics-400 [25] as described in [7]. We initialize the remaining layers of our model (eg. RPN, Action Transformer heads etc) from scratch, fix the running mean and variance statistics of batch norm layers to the initialization from the pre-trained model, and then finetune the full model end-to-end. Note that the only batch

norm layers in our model are in the I3D base and head networks; hence, no new batch statistics need to be estimated when finetuning from the pretrained models.

Data Augmentation: We augment our training data using random flips and crops. We find this was critical, as removing augmentation lead to severe overfitting and a significant drop in performance. We evaluate the importance of pretraining and data augmentation in Section 4.6.

SGD Parameters: The training is done using synchronized SGD over V100 GPUs with an effective batch size of 30 clips per gradient step. This is typically realized by a per-GPU batch of 3 clips, and total of 10 replicas. However, since we keep batch norm fixed for all experiments except for from-scratch experiments, this batch size can be realized by splitting the batch over 10, 15 or even 30 replicas for our heavier models. Most of our models are trained for 500K iterations, which takes about a week on 10 GPUs. We use a learning rate of 0.1 with cosine learning rate annealing over the 500K iterations, though with a linear warmup [14] from 0.01 to 0.1 for the first 1000 iterations. For some cases, like models with Action Transformer head and using ground truth boxes (Section 4.2), we stop training early at 300K iterations as it learns much faster. The models are trained using standard loss functions used for object detection [20], except for sigmoid cross-entropy for the multi-label classification loss.

4. Experiments

In this section we experimentally evaluate the model on the AVA benchmark. We start with introducing the dataset and evaluation protocol in Section 4.1. Note that the model is required to carry out two distinct tasks: action *localization* and action *classification*. To better understand the challenge of each independently, we evaluate each task given perfect information for the other. In Section 4.2, we replace the RPN proposals with the groundtruth (GT) boxes, and keep the remaining architecture as is. Then in Section 4.3, we assume perfect classification by converting all class labels into a single 'active' class label, reducing the problem into a pure 'active person' vs background detection problem, and evaluate the person localization performance. Finally we put the lessons from the two together in Section 4.4. We perform all these ablative comparisons on the AVA validation set, and compare with the state of the art on the test set in Section 4.5.

4.1. The AVA Dataset and Evaluation

The Atomic Visual Actions (AVA) v2.1 [15] dataset contains 211K training, 57K validation and 117K testing clips, taken at 1 FPS from 430 15-minute movie clips. The center frame in each clip is exhaustively labeled with all the person bounding boxes, along with one or more of the 80 action classes active for each instance. Following previous works [15, 41], we report our performance on the subset of 60 classes that have at least 25 validation examples. For comparison with other challenge submissions, we also report the performance of our final model on the test set, as reported from the challenge server. Unless otherwise specified, the evaluation is performed using frame-level mean average precision (frame-AP) at IOU threshold of 0.5, as described in [15].

Trunk	Head	QPr	GT Boxes	Params (M)	GFlops	Val mAP
I3D	I3D	-		16.2	6.5	21.3
I3D	I3D	-	✓	16.2	6.5	23.4
I3D	Tx	LowRes		13.9	33.2	17.8
I3D	Tx	HighRes		19.3	39.6	18.9
I3D	Tx	LowRes	✓	13.9	33.2	29.1
I3D	Tx	HighRes	✓	19.3	39.6	27.6

Table 1: **Action classification with GT person boxes.** To isolate classification from localization performance, we evaluate our models when assuming groundtruth box locations are known. It can be seen that the Action Transformer head has far stronger performance than the I3D head when GT boxes are used. All performance reported with $R = 64$ proposals. To put the complexity numbers into perspective, a typical video recognition model, 16-frame R(2+1)D network on Kinetics, is 41 GFlops [42]. For a sense of random variation, we retrain the basic Tx model (line 5) three times, and get a std deviation of 0.45 (on an mAP of 29.1).

4.2. Action classification given GT person boxes

In this section we assess how well the head can classify the actions, given the ground truth bounding boxes provided with the AVA dataset. This will give an upper bound on the action classification performance of the entire network, as the RPN is likely to be less perfect than ground truth. We start by comparing the I3D head with and without GT boxes in Table 1. We use a lower value of $R = 64$ for the RPN, in order to reduce the computational expense of these experiments. It is interesting to note that we only get a small improvement by using groundtruth (GT) boxes, indicating that our model is already capable of learning a good representation for person detection. Next, we replace the I3D head architecture with the Action Transformer, which leads to a significant 5% boost for the GT boxes case. It is also worth noting that our Action Transformer head implementation actually has 2.3M fewer parameters than the I3D head in the LowRes QPr case, dispelling any concerns that this improvement is simply from additional model capacity. The significant drop in performance with and without GT boxes for the Action Transformer is due to only using $R = 64$ proposals. As will be seen in subsequent results, this drop is eliminated when the full model with $R = 300$ proposals is used.

RoI source	QPr	Head	Val mAP	
			IOU@0.5	IOU@0.75
RPN	-	I3D	92.9	77.5
RPN	LowRes	Tx	77.5	43.5
RPN	HighRes	Tx	87.7	63.3

Table 2: **Localization performance (action agnostic).** We perform classification-agnostic evaluation to evaluate the performance of the heads for person detection. We observe that the I3D head is superior to Action Transformer-head model, though using the HighRes query transformation (QPr) improves it significantly. All performance reported with $R = 64$ proposals.

Head	QPr	#proposals	Val mAP
I3D	-	64	21.3
I3D	-	300	20.5
Tx	HighRes	64	18.9
Tx	HighRes	300	24.4
Tx+I3D	HighRes	300	24.9

Table 3: **Overall performance.** Putting the Action Transformer head with HighRes preprocessing and 300 proposals leads to a significant improvement over the I3D head. Using both heads: I3D for regression and Tx for classification performs best.

4.3. Localization performance (action agnostic)

Given the strong performance of the Action Transformer for the classification task, we look now in detail to the localization task. As described previously, we isolate the localization performance by merging all classes into a single trivial one. We report performance in Table 2, both with the standard 0.5 IOU threshold, and also with a stricter 0.75 IOU threshold.

The I3D head with RPN boxes excels on this task, achieving almost 93% mAP at 0.5 IOU. The naive implementation of the transformer using a low-resolution query does quite poorly at 77.5%, but by adopting the high-resolution query, the gap in performance is considerably reduced (92.9% to 87.7%, for the IOU-0.5 metric). The transformer is less accurate for localization and this can be understood by its more global nature; additional research on this problem is warranted. However as we will show next, using the HighRes query we can already achieve a positive trade-off in performance and can leverage the classification gains to obtain a significant overall improvement.

4.4. Putting things together

Now we put the transformer head together with the RPN base, and apply the entire network to the tasks of detection and classification. We report our findings in Table 3. It can be seen that the Action Transformer head is far superior to the I3D head (24.4 compared to 20.5). An additional boost can be obtained (to 24.9) by using the I3D head for regression and the Action Transformer head for classification – reflecting their strengths identified in the previous sections – albeit at a slightly higher (0.1GFlops) computational overhead.

Method	Modalities	Architecture	Val mAP	Test mAP
Single frame [15]	RGB, Flow	R-50, FRCNN	14.7	-
AVA baseline [15]	RGB, Flow	I3D, FRCNN, R-50	15.6	-
ARCN [41]	RGB, Flow	S3D-G, RN	17.4	-
Fudan University	-	-	-	17.16
YH Technologies [51]	RGB, Flow	P3D, FRCNN	-	19.60
Tsinghua/Megvii [22]	RGB, Flow	I3D, FRCNN, NL, TSN, C2D, P3D, C3D, FPN	-	21.08
Ours (Tx-only head)	RGB	I3D, Tx	24.4	24.30
Ours (Tx+I3D head)	RGB	I3D, Tx	24.9	24.60
Ours (Tx+I3D+96f)	RGB	I3D, Tx	**25.0**	**24.93**

Table 4: **Comparison with previous state of the art and challenge submissions.** Our model outperforms the previous state of the art by $> 7.5\%$ on the validation set, and the CVPR'18 challenge winner by $> 3.5\%$ on the test set. We do so while only using a single model (no ensembles), running on raw RGB frames as input. This is in contrast to the various previous methods listed here, which use various modalities and ensembles of multiple architectures. The model abbreviations used here refer to the following. R-50: ResNet-50 [17], I3D: Inflated 3D convolutions [7], S3D(+G): Separable 3D convolutions (with gating) [48], FRCNN: Faster R-CNN [33], NL: Non-local networks [46], P3D: Pseudo-3D convolutions [32], C2D [42], C3D [42], TSN: Temporal Segment Networks [45] RN: Relation Nets [34], Tx: Transformer [31, 43] and FPN: Feature Pyramid Networks [28]. Some of the submissions also attempted to use other modalities like audio, but got lower performance. Here we compare with their best reported numbers.

4.5. Comparison with existing state of the art

Finally, we compare our models to the previous state of the art on the test set in Table 4. We find the Tx+I3D head obtains the best performance, and simply adding temporal context at test time (96 frames compared to 64 frames at training) leads to a further improvement. We outperform the previous state of the art by more than 7.5% absolute points on validation set, and the CVPR 2018 challenge winner by more than 3.5%. It is also worth noting that our approach is much simpler than most previously proposed approaches, especially the challenge submissions that are ensembles of multiple complex models. Moreover, we obtain this performance only using raw RGB frames as input, while prior works use RGB, flow, and in some cases audio as well.

4.6. Ablation study

All our models so far have used class agnostic regression, data augmentation and Kinetics [25] pre-training, techniques we observed early on to be critical for good performance on this task. We now validate the importance of those design choices. We compare the performance using the I3D head network as the baseline in Table 5. As evident from the table, all three are crucial in getting strong performance. In particular, class agnostic regression is an important contribution. While typical object detection frameworks [16, 20] learn a separate regression layers for each

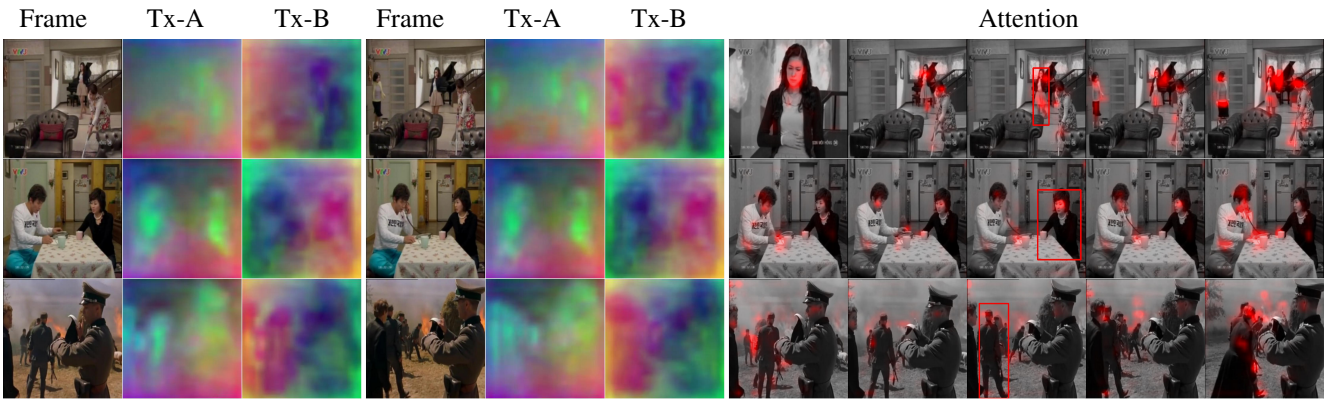

Figure 4: **Embedding and attention.** For two frames, we show their 'key' embeddings as color-coded 3D PCA projection for two of the six heads in our 2-head 3-layer Tx head. It is interesting to note that one of these heads learns to track people semantically (Tx-A: all upper bodies are similar color – green), while the other is instance specific (Tx-B: each person is different color – blue, pink and purple). In the following columns we show by the average softmax attention corresponding to the person in the red box for all heads in the last Tx layer. Our model learns to hone in on faces, hands and objects being interacted with, as these are most discriminative for recognizing actions.

	I3D head	Cls-specific bbox-reg	No Data Aug	From Scratch
Val mAP	21.3	19.2	16.6	19.1

Table 5: **Augmentation, pre-training and class-agnostic regression.** We evaluate the importance of certain design choices such as class agnostic box regression, data augmentation and Kinetics pre-training, by reporting the performance when each of those is removed from the model. We use the I3D head model as the baseline. Clearly, removing any leads to a significant drop in performance. All performance reported with $R = 64$ proposals.

object category, it does not make sense in our case as the 'object' is always a human. Sharing those parameters helps classes with few examples to also learn a good person regressor, leading to an overall boost. Finally, we note the importance of using a sufficient number of proposals in the RPN. As can be seen in Table 3, reducing the number from 300 to 64 decreases performance significantly for the Action Transformer model. The I3D head is less affected. It is interesting because, even for 64, we are using far more proposals than the actual number of people in the frame.

5. Analysis

We now analyze the Action Transformer model. Apart from obtaining superior performance, this model is also more interpretable by explicitly encoding bottom up attention. We start by visualizing the key/value embeddings and attention maps learned by the model. Next we analyze the performance vis-a-vis specific classes, person sizes and counts; and finally visualize common failure modes.

Learned embeddings and attention: We visualize the 128D 'key' embeddings and attention maps in Figure 4. We visualize the embeddings by color-coding a 3D PCA projection. We show two heads out of the six in our 2-head 3-layer Action Transformer model. For attention maps, we visualize the average softmax attention over the 2 heads in the last layer of our Tx head. It is interesting to note that our model learns to track the people over the clips, as shown from the embeddings where all 'person' pixels are same color. Moreover, for the first head all humans have the same color, suggesting a *semantic* embedding, while the other has different, suggesting an *instance-level* embedding. Similarly, the softmax attention maps learn to attend and track faces, hands and other parts of the person of interest as well as the other people in the scene. It also tends to attend to objects the person interacts with, like the vaccum cleaner and coffee mugs. This makes sense as many actions in AVA such as talking, listening, hold an object etc. require focusing the faces, hands of people and objects to deduce.

Breaking down the performance: We now break down the performance of our model into certain bins. We start by evaluating the performance per class in Figure 5 (a). We sort the performance according the increasing amounts of training data, shown in green. While there is some correlation between the training data size and performance, we note that there exist many classes with enough data but poor performance, like smoking. We note that we get some of the largest improvement in classes such as sailing boat, watching TV etc, which would benefit from our Action Transformer model attending to the context of the person. Next, we evaluate the performance with respect to the size of the person in the clip, defined by the percentage area occupied by the GT box, in Figure 5 (b). For this, we split the validation set into bins, keeping predictions and GT within certain size limits. We find the size thresholds by sorting all the GT boxes and splitting into similar sized bins, hence ensuring

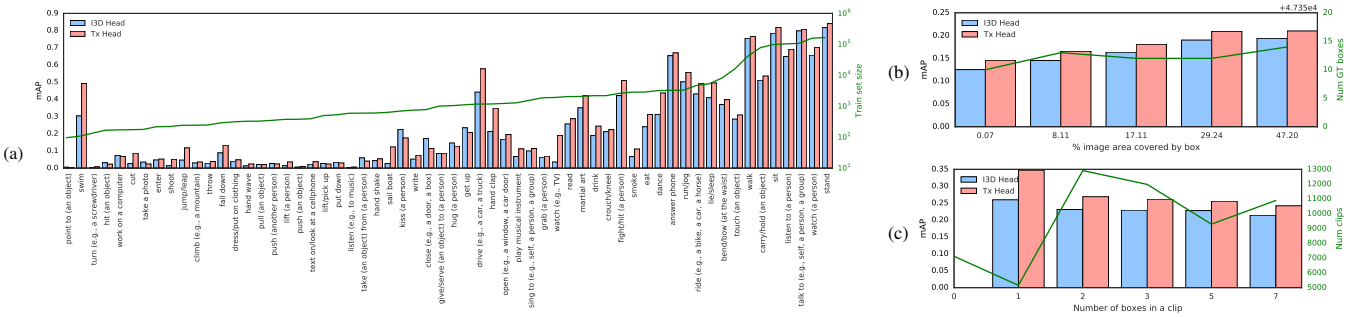

Figure 5: **Performance by (a) class, (b) box area and (c) count.** While overall trend suggests a positive correlation of performance with train-set size (green line), there do exist interesting anomalies such as 'smoking', 'eating' etc, which are still hard to recognize despite substantial training data. In (b) and (c) the green line denotes the validation subset size. We observe the performance largely improves as the person box size increases, and as number of boxes decreases. *Axis labels best viewed zoomed in on screen.*

Figure 6: **Top predictions.** Example top predictions for some of the classes using our model. Note that context, such as other people or objects being interacted with, is often helpful for the classifying actions like 'watching a person', 'holding an object' and so on. Capturing context is a strength of our model.

Figure 7: **Misclassified videos.** Videos from the 'smoking' class that obtains low performance even with large amount of training data. Failure modes include, (a) *Similar action/interaction:* In the first clip, the person has his hand on his mouth, similar to a smoker; and in the second, the mic looks like a cigarette; (b) *Identity:* There are multiple people (or reflections) and the action is not being assigned to the correct person; (c) *Temporal position:* The dataset expects the action to be occurring in the key frame, in these examples the action has either finished or not started by the key frame.

similar 'random' performance for each bin. We find performance generally increases with bigger boxes, presumably because it becomes progressively easier to see what the person is doing up close. Finally, we evaluate the performance with respect to the number of GT boxes labeled in a clip in Figure 5 (c). We find decreasing performance as we add more people in a scene.

Qualitative Results: We visualize some successes of our model in Figure 6. Our model is able to exploit the context to recognize actions such as 'watching a person', which are inherently hard when just looking at the actor. Finally, we analyze some common failure modes of our best model in Figure 7. The columns show some common failure modes like (a) similar action/interaction, (b) identity and (c) temporal position. A similar visualization for all classes is provided in the arXiv version [9].

6. Conclusion

We have shown that the Action Transformer network is able to learn spatio-temporal context from other human actions and objects in a video clip to localize and classify human actions. The resulting embeddings and attention maps (learned indirectly as part of the supervised action training) have a semantic meaning. The network exceeds the state-of-the-art on the AVA dataset by a significant margin. It is worth noting that previous state-of-the-art networks have used a motion/flow stream in addition to RGB [7, 48], so adding flow as input is likely to boost performance also for the Action Transformer network. Nevertheless, performance is far from perfect, and we have suggested several avenues for improvement and investigation.

Acknowledgements: We would like to thank V. Patraucean, R. Arandjelović, J.-B. Alayrac, A. Arnab, M. Malinowski and C. McCoy for helpful discussions.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016. 2

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *Stat*, 2016. 4

[3] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori. Object level visual reasoning in videos. In *ECCV*, 2018. 2

[4] F. Baradel, C. Wolf, and J. Mille. Human action recognition: Pose-based attention draws focus to hands. In *ICCV Workshop*, 2017. 2

[5] F. Baradel, C. Wolf, and J. Mille. Human activity recognition with pose-driven attention to rgb. In *BMVC*, 2018. 2

[6] F. Caba et al. Activitynet leaderboard. spatio-temporal action localization (ava-1. computer vision only). http://activity-net.org/challenges/2018/evaluation.html. 2

[7] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 3, 4, 6, 8

[8] K. Duarte, Y. S. Rawat, and M. Shah. VideoCapsuleNet: A simplified network for action detection. In *NIPS*, 2018. 2

[9] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. *arXiv preprint arXiv:1812.02707*, 2018. 4, 8

[10] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran. Detect-and-Track: Efficient Pose Estimation in Videos. In *CVPR*, 2018. 2, 4

[11] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *NIPS*, 2017. 2

[12] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017. 2

[13] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *TPAMI*, 2007. 2

[14] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 5

[15] C. Gu, C. Sun, D. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018. 2, 5, 6

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 2, 6

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[18] R. Herzig, E. Levi, H. Xu, E. Brosh, A. Globerson, and T. Darrell. Classifying collisions with spatio-temporal action graph networks. *arXiv preprint arXiv:1812.01233*, 2018. 2

[19] R. Hou, C. Chen, and M. Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *ICCV*, 2017. 2

[20] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017. 2, 4, 5, 6

[21] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 2

[22] J. Jiang, Y. Cao, L. Song, S. Zhang, Y. Li, Z. Xu, Q. Wu, C. Gan, C. Zhang, and G. Yu. Human centric spatio-temporal action localization. Technical report, Tsinghua University, 2018. 2, 6

[23] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017. 2

[24] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[25] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 2, 4, 6

[26] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 2

[27] I. Laptev. On space-time interest points. *IJCV*, 2005. 2

[28] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6

[29] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen. Attention clusters: Purely attention based local feature integration for video classification. In *CVPR*, 2018. 2

[30] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv:1706.06905*, 2017. 2

[31] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, and A. Ku. Image transformer. In *ICML*, 2018. 3, 4, 6

[32] Z. Qiu, T. Yao, and T. Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 6

[33] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 3, 6

[34] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017. 2, 6

[35] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004. 2

[36] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. *ICLR-Workshops*, 2016. 2

[37] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017. 2

[38] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2

[39] G. Singh, S. Saha, M. Sapienza, P. Torr, and F. Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *ICCV*, 2017. 2

[40] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, 2012. 2

[41] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, and C. Schmid. Actor-centric relation network. In

ECCV, 2018. 2, 5, 6

[42] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 5, 6

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017. 1, 2, 3, 4, 6

[44] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2

[45] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2, 6

[46] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 2, 6

[47] X. Wang and A. Gupta. Videos as space-time region graphs. In *ECCV*, 2018. 2

[48] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning for video understanding. In *ECCV*, 2018. 2, 6, 8

[49] H. Xu, A. Das, and K. Saenko. R-C3D: Region convolutional 3D network for temporal activity detection. In *ICCV*, 2017. 2

[50] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 2

[51] T. Yao and X. Li. YH Technologies at ActivityNet Challenge 2018. *arXiv preprint arXiv:1807.00686*, 2018. 2, 6