Hand Guided High Resolution Feature Enhancement for Fine-Grained Atomic Action Segmentation within Complex Human Assemblies – Supplementary Material –

1. Hand Localisation

1.1. Model details

In order to detect the presence and location of an assemblers hand within an image we train a mobilentv2 [4] model to output a fixed Sigmoid normalised vector of length 6 containing the elements $[\hat{P}_1, \hat{x}_1, \hat{y}_1, \hat{P}_2, \hat{x}_2, \hat{y}_2]$ where $\hat{P}_1$ and $\hat{P}_2$ are the probability of the left and right hand existing in the image (1 if hand present 0 otherwise) and $\hat{x}_i$, $\hat{y}_i$ are the normalised position of hand i in a given input image. We train with a modified mean squared error loss, shown in equation 1, that doesn't penalise the model for incorrectly predicting a hand position when no hand is present, where P_i is the (1,0) label of a present hand and x_i and y_i are the ground truth normalised location of hand i within a given image. λ is an equalising constant set to 0.1 to get similar learning rates across the loss function.

$$\mathcal{L} = \lambda \sum_{i=1}^{2} (P_i - \hat{P}_i)^2 + P_1[(x_1 - \hat{x}_1)^2 + (y_1 - \hat{y}_1)^2] + P_2[(x_2 - \hat{x}_2)^2 + (y_2 - \hat{y}_2)^2] \quad (1)$$

1.2. Implementation Details

Our hand localisation algorithms were trained with a learning rate of 0.04 for 200 epochs with a learning rate decrease at 100 and 150 epochs. With gradient clipping set to 20, parameter weight decay set to 0.0005 and momentum set to 0.9.

We train our hand model on 4531 randomly selected frames within our training data and test on a further 1919 frames. Roughly 15% of selected frames contain either just one or no hands. We labelled our hand data with a custom GUI which we also plan to make publicly available [1]

1.3. Hand Localisation Performance

In order to evaluate the performance of of hand localisation model we implement the following metric: $F1@T_L{>}X$. Under our model output we get a prediction per hand in the form $(\hat{P}, \hat{x}, \hat{y})$ where $\hat{P}$ is the probability of a hand being present within the frame, and $\hat{x}, \hat{y}$ are the normalised position of the hand within the frame. We allocate our predictions a true positive label if $\hat{P}$ is grater than 0.5 (*i.e.* there is more than 50% probability the hand is present) and the there is a ground truth hand present, and the spatial prediction of the hand location is within a certain threshold of the ground truth location, given by $\sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2} < T_L$, where T_L is a changeable threshold and x and y are the ground truth hand position. If a hand is predicted to exist but has the wrong location or there is no hand present then the prediction will be assigned false positive. A prediction is false negative when a hand not predicted but a hand is present within the image. We report the hand performance for various location thresholds T_L in table 1.

T_L	$F1@T_L$
0.05	74.5
0.1	83.0
0.2	92.5
0.3	96.8

Table 1. Hand localisation performance on our novel assembly dataset.

2. Hand Feature Spatial Alignment

Assuming an input image of size (W,H) and a down sample operation to size S for input into the back bone model, the relative resolution between the two inputs is (H/S). In order to achieve this spatial alignment between hand cutouts of size (w_c,h_c), and the backbone cutout of size C_s, we calculate the normalised size $(\overline{w}_{hand}, \overline{h}_{hand})$ via equation 2 and offsets $(\overline{x}_{hand}, \overline{y}_{hand})$ via equation 3 with respect to the input image to the backbone stream. The variables in these equations are defined in Fig. 1 which shows the down scaling of the image from the shortest side from H to S and random crop of size C_s at offset (x_c, y_c) (for regularisation during training) of the input frame.

[1] Removed for blind review

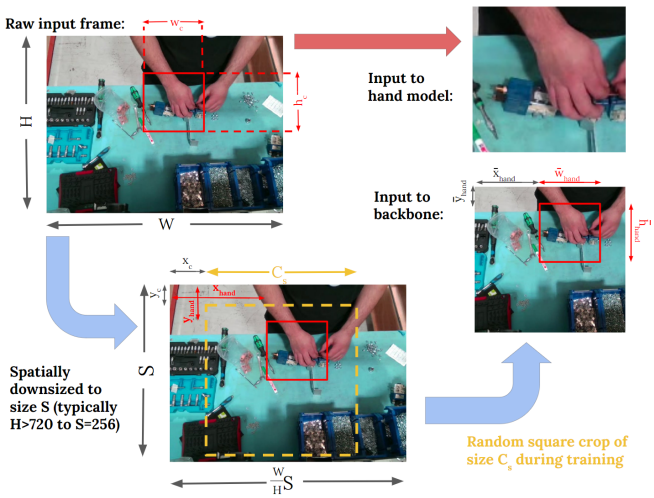

Figure 1. The location and size of the extracted hand image with respect to the input image to the backbone model after pre-processing.

$$\overline{w}_{hand} = \left(\frac{w_c}{H}\right)\left(\frac{S}{C_s}\right), \quad \overline{h}_{hand} = \left(\frac{h_c}{H}\right)\left(\frac{S}{C_s}\right) \quad (2)$$

$$\overline{x}_{hand} = \frac{x_{hand} - x_c}{C_s}, \quad \overline{y}_{hand} = \frac{y_{hand} - y_c}{C_s} \quad (3)$$

3. Model Implementation Details

All models were implemented within the Pytorch deep learning framework with a Resnet50 model used for the backbone architecture and a ResNet18 used for the hand model. All models utilised Kinetics-400[2] pretraining, with batch normalisation statistics frozen from pretrained weights to reduce overfitting. The initial learning rate was set to 0.002 for parameters within the backbone model and 0.0002 for parameters within the hand feature extraction model, with gradient clipping set to 20. Parameter weight decay was set to 0.0005 and momentum set to 0.9 for all parameters. All models were trained for 150 epochs with a batch size of 64 (except for the 16 frame model which utilised a batch size of 32), with learning rate decreased by a factor of 10 at 100 and 125 epochs. A dropout of 0.8 was implemented in the final fully connected layer of all model to reduce overfitting.

Models were trained on the a GPU node of the Hartree Centre Jade-2 HPC [2], utilising a single Tesla V-100 GPU with a wall clock training time of $\sim$ 6 hours per model.

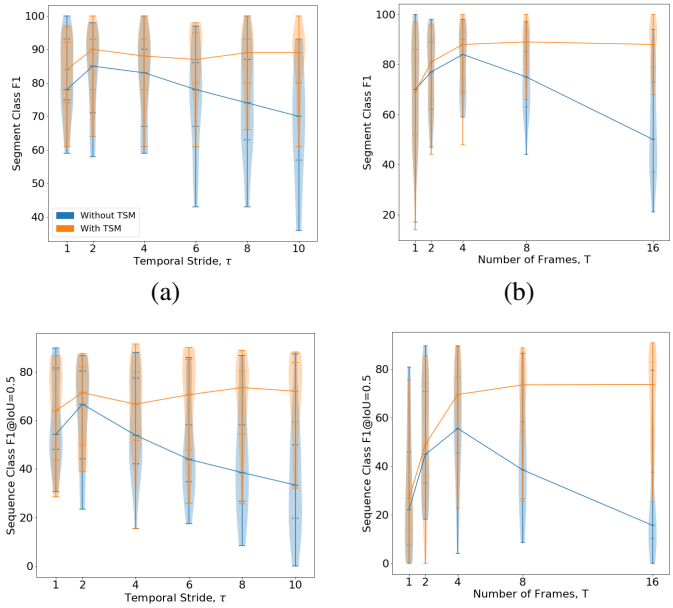

Figure 2. **Importance of temporal reasoning in the backbone model.** All graphs show the F1 class distributions for a temporal segment network [5] with and without inserted temporal shift modules[3] varying either the temporal stride, τ, with fixed T=8 (a and c), or varying number of input frames, T, with fixed τ=8 (b and d) on either a segment (a and b) or on an extended sequence (c and d) level after temporally aware label cleaning.

4. Importance of Temporal learning

In order to investigate the importance of temporal reasoning we train a set of identical Temporal Segment Networks (TSN) with and without inserted TSM with varying numbers of input frames and temporal stride between input frames, with results shown in Fig. 2.

We first fix T to the commonly used 8 frames [1, 3] and vary the temporal stride $\tau \in \{1,2,4,6,8,10\}$ with results shown in Fig. 2 (a and c on a segment and sequence level respectively). It is clear that temporally aware TSM models outperform standard TSN models across all temporal strides, and maintain a high F1 score beyond $\tau = 2$, while the performance of a regular TSN model, incapable of temporal learning, drops as τ increases, suggesting sparser frames can introduce more useful long term information for classification, but only when temporal learning is possible. This drop in performance is more significant on a sequence level with the performance of some classes dropping to an F1 score of 0 at $\tau = 10$, suggesting sparser frames when temporal learning is not applicable leads to significant confusion when operating in a sliding window fashion.

Secondly, we vary the number of input frames

[2] https://www.jade.ac.uk/

T$\in\{1,2,4,8,16\}$ while keeping $\tau = 8$. Fig. 2 (b and d on a segment and sequence level respectively) show that for both a TSN and TSM model increasing the number of input frames helps the models initially, however, beyond $T = 4$ the TSN model performance drops significantly, while the TSM model continues to perform well, suggesting again that providing more temporal information by using multiple frames at input is only useful if a temporally aware backbone model is used.

5. Single *vs.* 8 Frame - Segment Level

Figure 3 shows the class F1 improvements comparing a single frame and a temporally aware 8 frame model on a temporally cropped segment level. In keeping with analysis on a sequence level, temporally salient classes see the largest improvement, with place metal bar seeing the largest improvement from an F1 score of 17 to 79. It is noted that the improvements in class performance on a segment level is smaller than that on a sequence level as mentioned in section 1.4.4, suggesting temporal modelling is more imperative when operating on a sequence level, with no action seeing a much smaller improvement, suggesting when segments are neatly temporally cropped they are easier to distinguish on a spatial level.

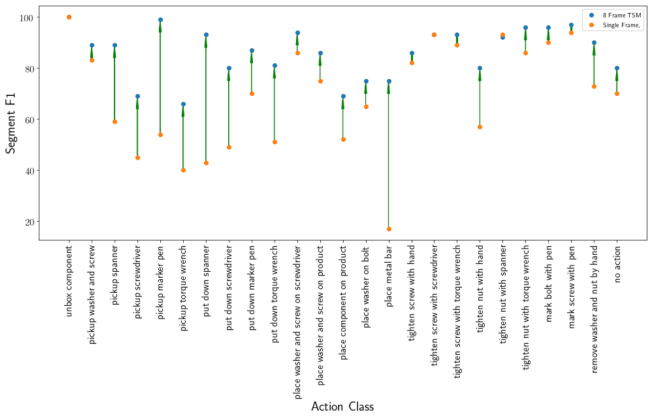

Figure 3. Improvement in $F1$ score across classes on a segment level when comparing a single frame to an 8 frame temporally aware TSM model.

6. High Resolution Hand model - Sequence Level

For completeness we also include the difference in class performance between a baseline TSM model and our high resolution hand model on sequence level. Figure 4 shows that there is virtually no change in no action performance on a sequence level, again suggesting high resolution hand features are improving the models ability to distinguish between classes rather than distinguish between an action and no action.

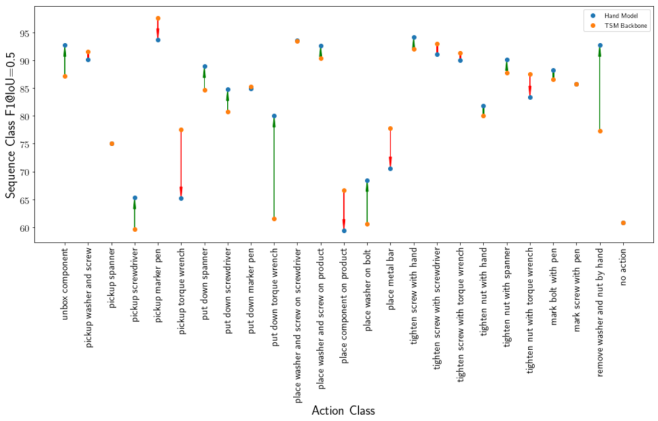

Figure 4. Improvement in $F1@IoU > 0.5$ score across classes on a sequence level when comparing an 8 frame temporally aware TSM model to an equivalent model with our high resolution hand enhanced features.

7. Model Prediction Visualisations

In the following section we produce two model prediction visualisations extracted from two extended unseen assembly sequences using an 8 frame TSM model with high resolution hand feature enhancement. Figure 5 and 6 show class colour coded predictions against ground truth predictions in grey, with black predictions representing incorrectly predicted frames.

As can be seen, despite the model operating in a sliding window fashion it is capable of accurately classify nearly all actions, predicting the start and end of actions successfully despite not being explicitly trained to do so. One notable challenging scenario encountered by the model is rapid changes from one action to another which are often separated by a few frames of "no action". This problem is highlighted in Figure 6 when the operator repeatedly places a screw on the product and then tightens the screw with their hand. In this situation the model is liable to miss many short no action segments - combining them into one of the rapidly changing actions, however can still accurately segment the respective actions with very good precision.

References

[1] Chun-Fu Chen, Rameswar Panda, Kandan Ramakrishnan, Rogerio Feris, John Cohn, Aude Oliva, and Quanfu Fan. Deep analysis of cnn-based spatio-temporal representations for action recognition, 2021.

[2] Will Kay, Joao 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, 2017.

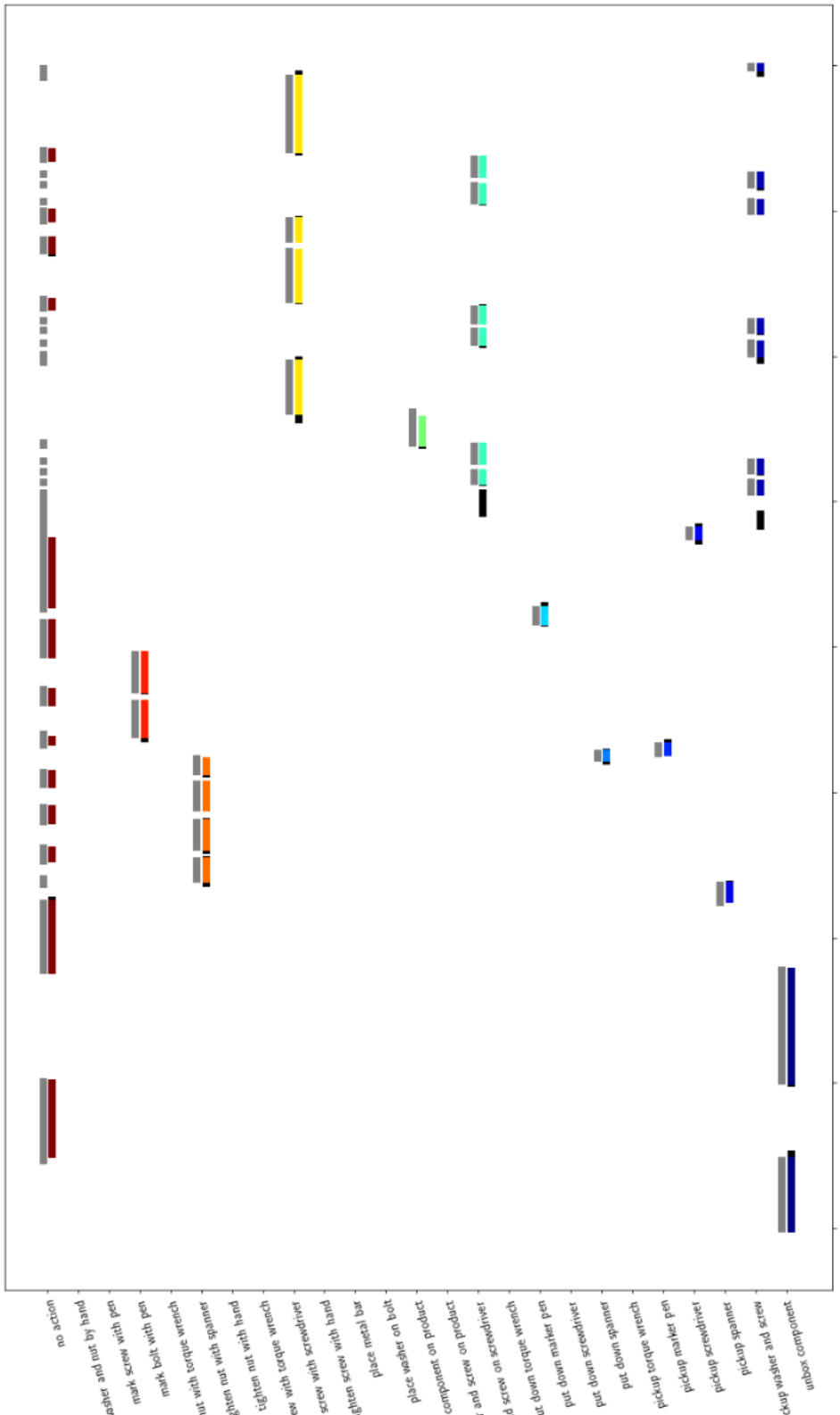

Figure 5. Model predictions on an extended video sequence.

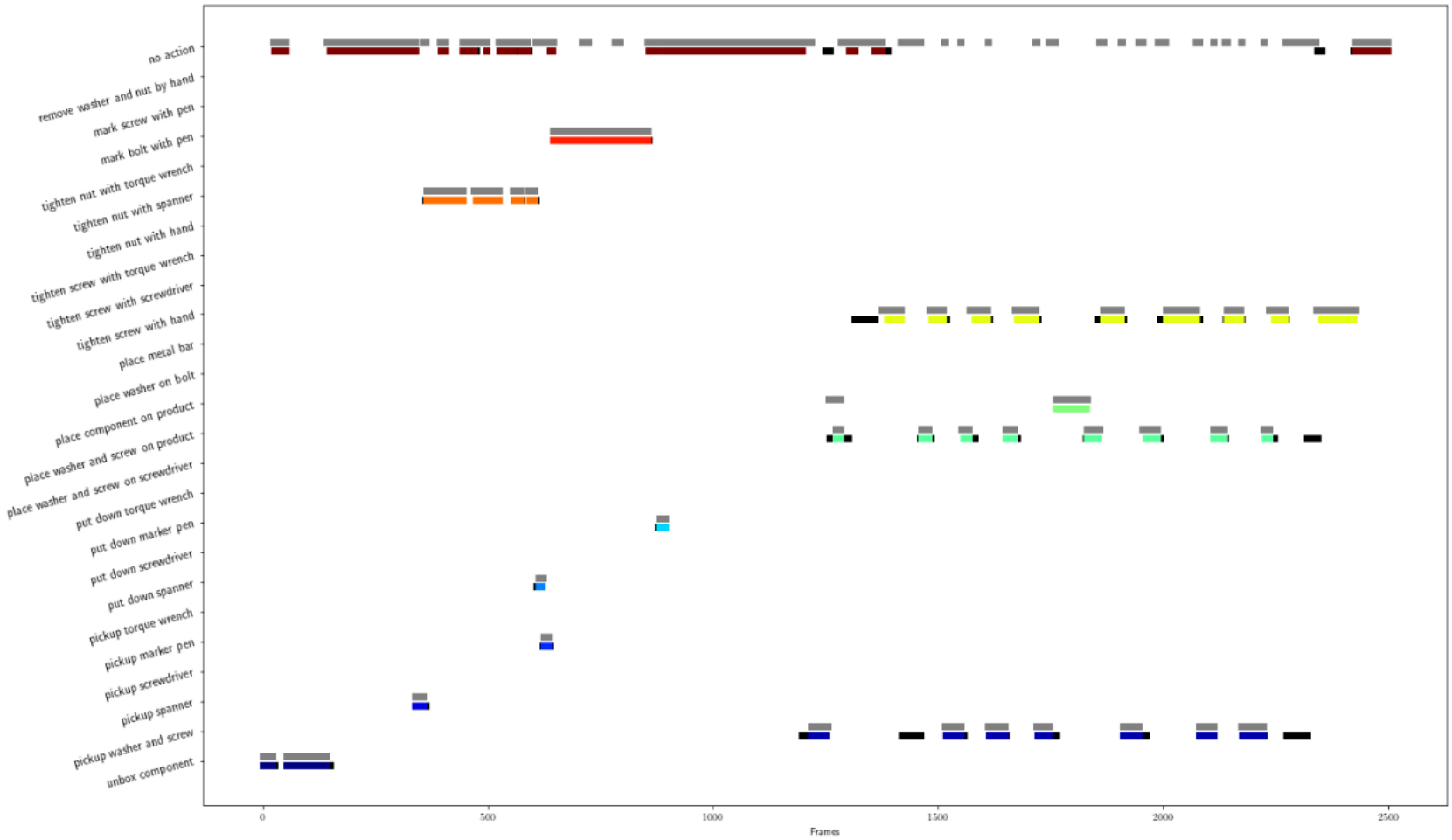

Figure 6. Model predictions on an extended video sequence.

[3] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[5] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 20–36, Cham, 2016. Springer International Publishing.