Recognising Actions for Instructional Training using Pose Information: A Comparative Evaluation

Seán Bruton and Gerard Lacey
Graphics, Vision and Visualisation (GV2), Trinity College Dublin, University of Dublin, Ireland

Keywords: Action Recognition, Deep Learning, Pose Estimation.

Abstract: Humans perform many complex tasks involving the manipulation of multiple objects. Recognition of the constituent actions of these tasks can be used to drive instructional training systems. The identities and poses of the objects used during such tasks are salient for the purposes of recognition. In this work, 3D object detection and registration techniques are used to identify and track objects involved in an everyday task of preparing a cup of tea. The pose information serves as input to an action classification system that uses Long-Short Term Memory (LSTM) recurrent neural networks as part of a deep architecture. An advantage of this approach is that it can represent the complex dynamics of object and human poses at hierarchical levels without the need for design of specific spatio-temporal features. By using such compact features, we demonstrate the feasibility of using the hyperparameter optimisation technique of Tree-Parzen Estimators to identify optimal hyperparameters as well as network architectures. The results of 83% recognition show that this approach is viable for similar scenarios of pervasive computing applications where prior scene knowledge exists.

1 INTRODUCTION

Humans perform many complex tasks with their hands. These tasks often involve manipulating objects in a specific manner to achieve a goal. A way in which humans learn the motor skills necessary to perform these tasks is by repetition with evaluative feedback from an expert supervisor (Debarnot et al., 2014; Ericsson et al., 1993). A formalised method of this type of training, known as Direct Observation of Procedural Skills (DOPS), has proven effective for improving undergraduate medical skills (Profanter and Perathoner, 2015). This observational training has drawbacks however. These include the subjectivity and biases of the supervisor, the logistical requirement of the physical presence of the supervisor, and the cost associated with providing supervisors for all students. To overcome these issues, we wish to develop techniques to automate the supervisory role in learning physical tasks. A key challenge of implementing such a system is the recognition of the individual actions that are part of the task being performed. In this work, we use an example task of preparing a cup of tea in place of a medical skill to explore methods of constructing such a system.

Using fixed arrangements for the system allows greater flexibility in types of camera sensors. The extra depth information available from consumer RGB-D cameras has been used to significant advantage in tackling the problem of tracking humans at various granularities (Shotton et al., 2013; Tang et al., 2016). Another area where the availability of depth data has improved results is object pose estimation (Hinterstoisser et al., 2012). These continuing advances could allow for accurate real-time tracking of objects and people, providing valuable input information for a system that determines if a complex object manipulation task has been performed correctly.

The problem remains, however, of how to use this information to understand the interactions between people and objects. In this work, we use recurrent neural networks to recognise human-object interactions based on tracked object poses. Training against compact pose data permits training to be completed in a reasonable timeframe. We use this property to perform a hyperparameter search for architectural and algorithmic parameters, automating the costly process of identifying an optimal architecture.

As part of our work, we also make available the dataset used to evaluate this system. This dataset comprises performances of an activity recorded with a multi-camera RGB-D set-up, as well as 3D scans of all the objects used (https://www.scss.tcd.ie/gerard.lacey).

Bruton, S. and Lacey, G.
Recognising Actions for Instructional Training using Pose Information: A Comparative Evaluation.
DOI: 10.5220/0007395304820489
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 482-489
ISBN: 978-989-758-354-4

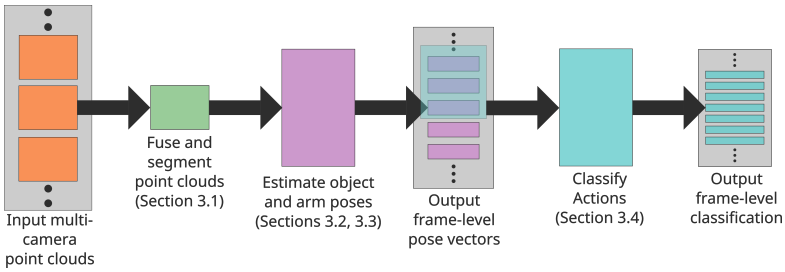

Figure 1: The high-level overview of the stages of the action recognition system.

2 RELATED WORK

Our task is to recognise a sequence of human-object interactions as part of a fixed activity. In the literature, systems of providing situational support for such goal-directed activities have been developed. Much of these systems make use of inertial sensors, such as accelerometers, attached to the objects in use. Sensor information has been combined with image features to train discriminative classifiers, such as random forests, to recognise actions involved in preparation of a salad (Stein and McKenna, 2013). A drawback of sensor-based approaches is that the presence of sensors is less natural for users and may hinder them in performing the task using their typical technique.

Other methods for recognition of actions as part of a complex task have focussed on the tracking of objects and the design of features based on associated motion patterns. On a dataset of cooking activities, a number of object tracking methods were tested in performing fine-grained action recognition (Rohrbach et al., 2012). It was found that a pose-based descriptor approach, based on Fourier transform features, underperformed relative to dense trajectories (Wang et al., 2011). Other authors (Stein and McKenna, 2017) attempt to recognise component fine-grained actions of the complex activity of preparing a salad by devising a custom feature descriptor based on histograms of tracklets described relative to the object in use.

An alternative to tracking individual objects, other approaches use global image features to perform segmentation and recognition across an entire activity sequence. One recent work (Kuehne et al., 2016) used a generative framework to segment fine-grained activities. Similarly based on language and grammar models, other authors (Richard and Gall, 2016) use a probabilistic model that models the segmentation and classification of actions jointly. However, due to these segmental approaches requiring observation of the entire sequence, it would not be possible to provide supervisory feedback during an activity.

The resurgence of neural networks has seen CNNs used to produce global image features. A recent work (Lea et al., 2016) utilised a CNN to extract image features, and used a 1D convolution over the output image features to classify fine-grained actions of goal-directed activities. The authors also utilise a semi-Markov model to segment a performance video into actions, however this improves accuracy marginally for the 50 Salads dataset (Stein and McKenna, 2013).

This variety of approaches shows that there is no consensus on the best techniques of recognising actions for situational support systems.

3 SYSTEM DESIGN

Here, we detail the component stages of the entire recognition pipeline. The pipeline itself is illustrated in Figure 1, with references to the following sections.

3.1 Sensor Fusion

For the task performance, we assume that it can be performed on the surface of a table. To eliminate occlusions of salient information for action recognition we use an array of three RGB-D cameras, arranged as per Figure 2. For each camera, we construct point cloud representations from each RGB-D frame.

It is necessary to synchronize the point clouds received from the different cameras to allow for fusion into a single merged point cloud. To do so, we associate point clouds by inspecting the time differences between receipt of the data from the different cameras and determine if the difference is within a specified threshold.

Interference is an undesirable side effect of using multiple structured infra-red sensors to observe the same target. This results in holes and noise in the depth maps. The “Shake ‘N’ Sense” technique (Butler et al., 2012) is used to address this problem. This technique involves vibration of affected sensors which causes the patterns of other cameras to appear blurred relative to its own pattern. The advantages of

Figure 2: The arrangement of the recording set-up. Top-left image shows the vibration motor attached to an RGB-D camera (Asus Xtion). Right image shows the placement of the cameras in relation to the task table. Bottom-left image shows a colour image taken from the centre camera during a recording.

this technique is that it maximises the amount of data available and has zero computational cost.

To fuse the point clouds, we estimate the rigid transformation between camera pairs. The pipeline for the registration, is to detect discriminative local features in each of the point clouds, find matches for these features and estimate a registration based on these matches. Intrinsic Shape Signature (ISS) keypoints (Zhong, 2009) were determined for each sensor's point cloud and Signatures of Histograms of Orientations (SHOT) descriptors were calculated (Salti et al., 2014) at these keypoints. To provide highly discriminative features, a number of target objects with characteristic geometry were included in the scene. To calculate the rigid transformation, the system of linear equations is solved using Levenberg Marquardt. This alignment is refined using the Generalised Iterative Closest Point algorithm (Segal et al., 2009), which estimates a dense registration after the coarse feature-based registration.

We use the presence of the task table to extract from the merged cloud all points above the table, isolating interesting points for our purposes. The RANSAC algorithm is used to find the table in the point cloud (Schnabel et al., 2007). Once the largest plane has been identified, Euclidean clustering (Rusu, 2010) is used to isolate the table points from other points that lie on this 3D plane. To identify all the points above the table, we construct a convex hull around the table cluster and check whether a point lies within a volume extruded above this hull.

3.2 Object Pose Estimation

The technique for estimating object poses is composed of two stages: estimating the initial pose at the beginning of a video; and tracking the object through the remainder of the video. The first stage identifies an object on the task table using the Linemod technique (Hinterstoisser et al., 2012). This technique combines colour gradients and surface normals to generate templates for known objects which can be efficiently searched. This technique has the benefit of working for objects that may have little surface texture, or are partially occluded. The templates are collected by performing three dimensional scans of the objects, and calculating the templates of the objects at different possible poses.

Figure 3: A rendering of scanned objects as meshes.

The second stage tracks an object as the video progresses. As a result of the merging and segmenting of the point clouds, we have isolated all points that should only belong to either objects or subject's arms. Based on the understanding that objects move small distances between frames, poses are registered frame to frame using the Generalised Iterative Closest Point (GICP) algorithm (Segal et al., 2009). This algorithm uses a probabilistic model for a point to point cost function, which has been shown to be more robust to incorrect correspondences than other iterative closest point algorithms. Point clouds of scanned versions of the objects, for example those in Figure 3, are used for registration. To ensure consistent densities of points across the source and target point clouds, the points are filtered using a voxel grid of fixed cell dimensions.

Given the initial pose of the object, estimated using the Linemod algorithm, and the incremental poses, found via registration of the scanned version of an object transformed to the previous pose, a series of poses for each object for a set of merged point clouds will be produced. The poses are encoded as a transformation matrix, $\mathbf{T} : \mathbb{R}^4 \rightarrow \mathbb{R}^4$. In homogeneous form, this transformation matrix is composed of a rotation, $\mathbf{R} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ and a translation $\mathbf{t} \in \mathbb{R}^3$, $T = [\mathbf{R}, \mathbf{t}; \mathbf{0}^T, 1]$. A more dimensionally compact representation of a transformation can be found by using unit quaternions to encode the rotation. Thus each pose is encoded as $[\mathbf{t}, a, b, c, d]$, where the rotation is encoded by the unit quaternion $\mathbf{z} = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, $a, b, c, d \in \mathbb{R}$.

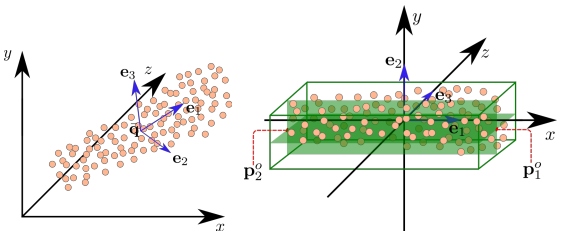

Figure 4: Left, a cluster centroid, $\bar{\mathbf{q}}$, and the three eigenvectors, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, is shown for an arm cluster. Right, the centroid is subtracted and the eigenvectors are axis-aligned. The centres of the two axis-aligned bounding box faces perpendicular to the principle axis, p_1^o and p_2^o, are used as the two end effectors for the arm pose.

3.3 Arm Pose Estimation

To estimate an arm pose, we use the segmented merged point cloud to identify arm points. Using the estimated object poses, we remove points that lie within a threshold distance of any points of the transformed object point clouds. Due to noise and slight misalignments of objects, it is necessary to perform further segmentation. Under the assumption that arm points all lie within a certain distance of each other, Euclidean clustering (Rusu, 2010) is used to identify the two largest clusters of the remaining points. A lower bound on possible cluster size ensures that only clusters large enough to be arms are selected.

We estimate the arm poses from the cluster information. We denote a candidate arm cluster as $Q = \{\mathbf{q}_i \in \mathbb{R}^3, i = 1, 2, \ldots\}$. We represent a pose using three components: $\mathbf{a}_1 \in \mathbb{R}^3$, the inner arm point in the task area; $\mathbf{a}_2 \in \mathbb{R}^3$, a point representing the position of the subject's wrist; and $r \in \mathbb{R}$, the width of the subject's arm as it appears to the central camera sensor. To calculate these features, the extents of an oriented bounding box over the cluster set Q are examined.

To determine these bounding box extents, Principle Component Analysis (PCA) is used. The centroid, $\bar{\mathbf{q}}$, is subtracted from each point, $\mathbf{q}_i$ in a candidate cluster. A data matrix, $\mathbf{X}$ is constructed, with each row representing a mean subtracted point. The eigenvalues of the matrix $\mathbf{X}^T\mathbf{X}$, are calculated along with the corresponding eigenvectors, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, representing the principle components of the cluster set. These eigenvectors can be composed into an orthogonal rotation matrix that transforms the three principle axes of greatest variance to the Cartesian axes, $\mathbf{R} = [\mathbf{e}_1^T, \mathbf{e}_2^T, \mathbf{e}_3^T]$. The new data matrix, $\mathbf{Y}^T = \mathbf{R}\mathbf{X}^T$, facilitates the determination of the bounding box extents. The maximal and minimal extents along each row of this matrix define the axis-aligned bounding box for the set of transformed arm points. The centres of the two faces perpendicular to the principle axis (see Figure 4), $\mathbf{p}_1^o$ and $\mathbf{p}_2^o$, can thus be found. Thus, the inverse transformation, $\mathbf{a}_i = \mathbf{R}^T\mathbf{p}_i^o + \bar{\mathbf{q}}$, can be applied to these points, to transform them back to the original arm cluster, to get the desired end effectors. The final feature component is the width of the observed arm cluster, r, corresponding to the bounding box width.

To distinguish between a left and right arm, the two clusters centroids are inspected. If a single cluster is detected, geometrical rules based on the position of the centroid and the angle of the arm are used to determine handedness. The threshold position and angle for these rules are defined based on the known scene arrangement.

3.4 Action Recognition

Given the individual feature components described in Sections 3.2 and 3.3, the feature vector is defined as the concatenation of the object features, $[\mathbf{t}_i, a_i, b_i, c_i, d_i]$ for each $i \in \{cup, pot, bowl, jug\}$, and the arm features, $[\mathbf{a}_{1,o}, \mathbf{a}_{2,o}, r_o]$ for each $o \in \{left, right\}$. Here, $\mathbf{a}_{1,o}$, $\mathbf{a}_{2,o}$, and r_o represent the inner point, outer point and width of the estimated pose of arm o, respectively. To capture temporal dynamics, a sequence of pose features is used for classification.

In the area of natural language processing, recurrent neural networks (RNNs) have proven useful in performing tasks such as machine translation and information modelling. More recent work, has looked at utilising recurrent neural networks to learn from sequences of image features extracted from videos (Donahue et al., 2016).

Long Short Term Memory Networks (LSTM) are an RNN structure designed to overcome the vanishing gradient problem (Hochreiter and Schmidhuber, 1997) by allowing for old (potentially useless) information to be forgotten and new (useful) information to be recorded in the cell state.

The LSTM cells are used as part of a larger neural network architecture to perform classification of sequences of pose features. The recognition performance is dependent upon the architecture. Decisions include whether to train LSTM cells on forward sequences or both forward and reverse sequences, illustrated by the 'reverse sequence' operation in Figure 5. Other decisions include the number of LSTM layers to use and the number of cell units per layer. These decisions are made using an automated procedure of identifying an optimal architecture, which will be discussed in Section 4.3.

In our architectures, the output of the LSTM layers undergoes further transformations via a series of fully connected layers. The output of the layers un-

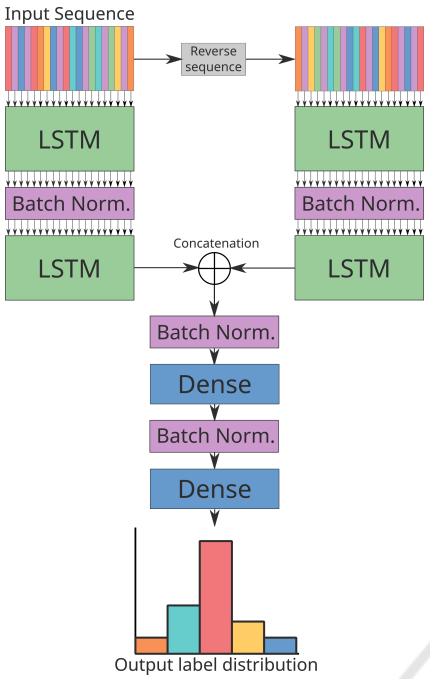

Figure 5: An example architecture used to classify sequences of pose information into action labels.

dergo an activation function, with intermediate dense layers outputs undergoing a Rectified Linear Unit (ReLU) activation. The outputs of the final layer undergo a softmax activation to produce an output distribution across the possible label classes.

Dropout is applied to the inputs of each of the dense layers for regularisation purposes. For the LSTM cells, a variant of dropout for recurrent neural networks is used, known as recurrent dropout (Gal and Ghahramani, 2016), whereby the dropout mask is identical across each of the timesteps in a sequence. An $\mathcal{L}_2$ loss is added based on the weights of the dense layers to prevent over dependence on singular neurons for classifications. Finally, batch normalisation is used between the layers of the network.

4 EXPERIMENTAL SET-UP

4.1 Dataset Collection

To evaluate our system, a dataset was collected of RGB-D videos of people performing the task of preparing a cup of tea. The dataset is composed of 24 samples recorded using three Asus Xtion Pro Live RGB-D cameras. A total of eight subjects were recorded performing the task three times. There were no restrictions imposed on how they prepared the cup of tea. The videos were all manually labelled with one of the five actions for each frame: 'pour tea'; 'pour milk'; 'add sugar'; 'stir'; and 'background'. A total of 25,913 frames were recorded, equating to an average video length of 38 seconds.

4.2 Training

We train each candidate neural network architecture to recognise the actions in this dataset. Given a sequence of pose features for a time t, $(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t)$, where $n \in \mathbb{N}$ and $\mathbf{x} \in \mathbb{R}^d$, and d is the length of the pose feature vector, we wish to estimate the output probability mass function, $\hat{p}_Y(y)$, where $y \in \mathcal{C}$, the set of action labels. Each neural network we train is a nonlinear differentiable function of the inputs, parametrised by the weights, $\hat{p}_Y(y) = \mathcal{F}(\mathbf{x}_{t-n}, \ldots, \mathbf{x}_t; \{\mathbf{W}_i\})$, where $\{\mathbf{W}_i\}$ is the set of weights contained in the network.

To train the network, back propagation is used. We minimise a loss function based on the categorical cross entropy, for each minibatch. The stochastic optimisation technique of Adam was used for this purpose.

Each network is trained for 250 epochs, with a minibatch size of 1024. The weights of the dense layers and LSTMs in the network were initialised with Xavier uniform initialisation, and the offset biases were initialised with zeros. The Tensorflow deep learning framework was used for implementation. We utilise a leave-one-subject-out cross validation, testing on an individual subject for each fold. This has the benefit of characterising the performance of the system for an unseen subject, identifying cases which the system may find challenging to classify correctly.

4.3 Hyperparameter Optimisation

The selection of optimal algorithm parameters can be time-consuming and involve expert knowledge that is difficult to convey. Due to elongated training times of neural networks, techniques such as grid search and random search can be infeasible. However, there exist methods of hyperparameter searching that can reduce the number of search iterations required. One approach utilises Tree-Structured Parzen Estimators (TPE) to model the target function, whereby each of the sampled points is represented with a Gaussian distribution in the hyperparameter space (Bergstra et al., 2011).

The next sample point is selected based on Expected Improvement. This can be defined as the expectation, under some model M of a fitness function $f : \mathcal{X} \to \mathbb{R}$, that $f(x)$ will negatively exceed some threshold y^*,

$$\mathbf{EI}_{y^*}(x) := \int_{-\infty}^{\infty} \max(y^* - y, 0) p_M(y|x) dy. \quad (1)$$

In our case, as well as model parameters, we wish to find optimal architectural parameters, such as number of LSTM layers, and whether to additionally train on reversed sequences (i.e. bidirectionally). Other hyperparameters searched over are shown in Table 1 as well as the prior distributions for each. The HyperOpt library (Bergstra et al., 2011) was utilised to perform this hyperparameter search.

Table 1: The prior distributions selected for the hyperparameter searches. $U(x,y)$ denotes the uniform distribution between x and y. Uniform distributions are used in all of the other discrete cases. These distributions are chosen based on a number of initial tests runs.

Parameter	*Prior*
Stride	1, 2 or 4
Sequence Length	8, 16, 32 or 64
Recurrent Units	64 or 128
Number LSTM layers	1, 2 or 3
Reverse sequences	True or False
Dense Kernel $\mathcal{L}2$	$U(0.0001, 0.01)$
LSTM input dropout rate	$U(0.0, 0.3)$
LSTM recurrent dropout rate	$U(0.0, 0.3)$
Softmax dropout rate	$U(0.0, 0.5)$
Initial learning rate	$U(0.0001, 0.01)$

4.4 Performance Benchmarking

To compare our recurrent neural network recognition approach, benchmark recognition algorithms are selected to compare against. These algorithms are Random Forest (Breiman, 2001) and Gradient Boosted Decision Trees (Friedman, 2001). These are selected due to their performance on high-dimensional recognition tasks (Shotton et al., 2013; Tang et al., 2016) and their use in related works (Stein and McKenna, 2017). The hyperparameter optimisation technique is also used to optimise the parameters of these classifiers. The hyperparameters that were searched over in the optimisation schedule were: the sequence stride; the sequence length; the number of tree estimators to use; and the maximum depth of the individual trees. In each optimisation, we maximise the F_1 score, calculated across all of the splits. This is calculated as the weighted harmonic mean of the precision and recall. For each classifier, fifty hyperparameter search iterations were performed, with each iteration tested using the cross-validation scheme.

5 RESULTS AND DISCUSSION

5.1 Object Pose Estimation

To evaluate the object pose estimation, a qualitative analysis is performed. We analyse the estimated poses and determine correctness based on their correspondence to the perceived object pose, from images as per Figure 6. Overall, the object pose estimation performs reliably under this analysis. Of the 24 recordings, with four objects tracked in each recording, there are two instances of an object's pose tracking being irrecoverably lost. In each instance, the object briefly becomes fully occluded during the performances by a combination of the subject's arms and other objects. There are four further instances where the alignment of an object's pose is perceptively incorrect for a portion of the recording. During the task performance, the cup is filled with tea which changes the perceived shape of the object, causing the registration algorithm to incorrectly identify correspondences. A potential method to overcome this issue would be to introduce a classification stage to determine whether the cup has become filled, and once it has, switch to registering an object scan of a filled cup.

5.2 Arm Pose Estimation

Similarly to the evaluation of object pose estimation, ground truth data for the arm pose estimation technique is unavailable and so we adopt a qualitative assessment methodology. Overall, the arm poses are estimated to an acceptable level for the majority of the performance videos. As the technique relies on a single merged cloud, errors do not accumulate. Observed errors include confusion between arms and inclusion of object parts or other body parts in the pose estimation, as shown in Figure 6. Much of these errors are caused by upstream errors in the object pose estimation stage. Other errors occur due to incorrect segmentations due to subjects moving arms close to their bodies.

5.3 Action Recognition

The optimal LSTM network uses temporal stride of a a single frame and a sequence length of 64. It contains one LSTM layer with 64 units for each sequence order. It performs better than the benchmark classifiers by a significant margin, as can be seen in Table 2. The accuracy is 82.9% calculated over all of the test splits. The F_1 metric, which we optimised against, is 81.72%, which is 8% above the other classifier results. The classifier also has a smaller standard devia-

Table 2: The classification metrics for the tested classifiers.

Classifier	Accuracy	Precision	Recall	F_1 score
Random Forest	76.21	82.54 ± 17.13	71.11 ± 20.59	73.55 ± 14.91
Gradient Boosting	75.68	**85.06** ± 18.12	70.45 ± 20.94	73.23 ± 13.89
LSTM	**82.90**	81.46 ± 8.65	**82.20** ± 7.62	**81.72** ± 7.49

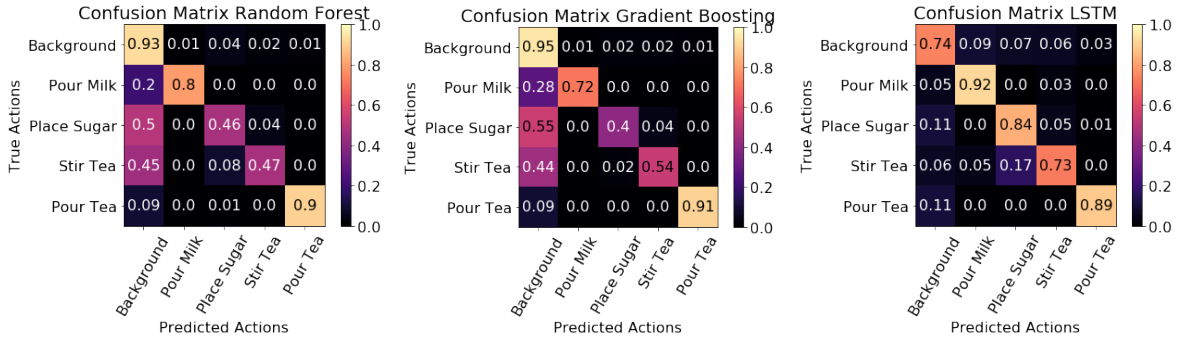

Figure 7: Confusion matrices for the three tested classification techniques. The LSTM classifier has more even distribution of correct predictions over all of the classes than the Random Forest or Gradient Boosting classifiers.

Figure 6: Pose estimation results for different sample recordings. In the first three rows, hand pose and object poses are estimated reliably. In the fourth row, object pose estimation errors are shown. These are due to misalignments of the cup due to changing topology and tracking losses due to complete occlusions. In the final row, arm pose estimation errors are shown. These errors are due to incorrect segmentation of left and right arm clusters when the hands are close together, and incorrect estimation based on inclusion of points from the subject's body.

tion across the action classes, indicating that it learns to discriminate actions more evenly.

The results indicate that the LSTM network is able to represent the dynamics of the human-object interactions to determine the current action. Analysing the confusion matrices, as shown in Figure 7, we observe that the actions *'Background'*, *'Place Sugar'* and *'Stir Tea'* are mistakenly predicted for each other. This may be more difficult to disambiguate as the arm pose dynamics present in these action are more subtle and do not necessarily involve the movement of an object.

To gain a deeper understanding of where the LSTM underperforms, we inspect its performance for individual test splits and observe that there is a large difference between the maximum F_1 score (93%) and minimum F_1 score (60%) for the splits. The worst performing splits contain the most significant pose estimation errors, as detailed in Sections 5.1 and 5.2. As such, improvements to these upstream methods should result in better performance for these splits.

6 CONCLUSIONS AND FUTURE WORK

In this work, we demonstrated a system that can classify human-object interactions for a goal-directed task to a high degree. For the classification method, we proposed the use of an optimised neural network architecture involving LSTMs. Analysing the results, we identified areas for further improvement in the pipeline and proposed potential methods to overcome these weaknesses. We release the multi-camera RGB-D video dataset of all task performances, including 3D scan data for each of the objects used. The system could potentially be applied to numerous real-world problems that require the understanding of human-object interactions, such as smart assembly line monitoring. We intend to further develop this system to

handle more complex interactions, such as those involved in procedural medical skills, as part of an automated instructional training system.

ACKNOWLEDGEMENTS

This research has been conducted under an Irish Research Council Enterprise Partnership Scholarship with Intel Ireland.

REFERENCES

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kgl, B. (2011). Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Butler, D. A., Izadi, S., Hilliges, O., Molyneaux, D., Hodges, S., and Kim, D. (2012). Shake'n'sense: Reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1933–1936, New York, NY, USA. ACM.

Debarnot, U., Sperduti, M., Di Rienzo, F., and Guillot, A. (2014). Experts bodies, experts minds: How physical and mental training shape the brain. *Frontiers in Human Neuroscience*, 8:280.

Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., and Darrell, T. (2016). Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ericsson, K. A., Krampe, R. T., and Tesch-Romer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review*, 100(3):363–406.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012). Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computing*, 9(8):1735–1780.

Kuehne, H., Gall, J., and Serre, T. (2016). An end-to-end generative framework for video segmentation and recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8.

Lea, C., Reiter, A., Vidal, R., and Hager, G. D. (2016). Segmental spatiotemporal cnns for fine-grained action segmentation. In *Computer Vision - ECCV 2016*, Lecture Notes in Computer Science, pages 36–52. Springer, Cham.

Profanter, C. and Perathoner, A. (2015). Dops (direct observation of procedural skills) in undergraduate skillslab: Does it work? analysis of skills-performance and curricular side effects. *GMS Zeitschrift fr Medizinische Ausbildung*, 32(4).

Richard, A. and Gall, J. (2016). Temporal action detection using a statistical language model. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3131–3140.

Rohrbach, M., Amin, S., Andriluka, M., and Schiele, B. (2012). A database for fine grained activity detection of cooking activities. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1194–1201.

Rusu, R. B. (2010). Semantic 3d object maps for everyday manipulation in human living environments. *KI - Knstliche Intelligenz*, 24(4):345–348.

Salti, S., Tombari, F., and Di Stefano, L. (2014). Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient ransac for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library.

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics: Science and Systems*, volume 2.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124.

Stein, S. and McKenna, S. J. (2013). Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 729–738, New York, NY, USA. ACM.

Stein, S. and McKenna, S. J. (2017). Recognising complex activities with histograms of relative tracklets. *Computer Vision and Image Understanding*, 154:82–93.

Tang, D., Chang, H., Tejani, A., and Kim, T. K. (2016). Latent regression forest: Structured estimation of 3d hand poses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1.

Wang, H., Klaser, A., Schmid, C., and Liu, C.-L. (2011). Action recognition by dense trajectories. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176.

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–696.