Recurrent Assistance: Cross-Dataset Training of LSTMs on Kitchen Tasks

Toby Perrett Dima Damen
Department of Computer Science, University of Bristol, UK
<firstname.lastname@bristol.ac.uk>

Abstract

In this paper, we investigate whether it is possible to leverage information from multiple datasets when performing frame-based action recognition, which is an essential component of real-time activity monitoring systems. In particular, we investigate whether the training of an LSTM can benefit from pre-training or co-training on multiple datasets of related tasks when it uses non-transferred visual CNN features. A number of label mappings and multi-dataset training techniques are proposed and tested on three challenging kitchen activity datasets - Breakfast, 50 Salads and MPII Cooking 2. We show that transferring, by pre-training on similar datasets using label concatenation, delivers improved frame-based classification accuracy and faster training convergence than random initialisation.

1. Introduction

Real-time monitoring of activities is essential for assistive technologies, to provide guidance, prompt reminders, raise alarms as well as provide input to higher-level routine observation methods. In this paper we focus on **frame-based action recognition** - the task of classifying the action taking place in each individual frame, which is a fundamental building block of real-time activity monitoring. Frame-based recognition poses significantly larger challenges than the more widely studied video-level action classification, which tends to be performed offline on temporally trimmed videos.

One of the main problems with the use of assistive visual monitoring systems in the wild is the requirement of a large amount of training data for each new environment, as models trained in one location tend not to generalise well to others. If improvements could be found, by leveraging existing data to circumvent or at least speed the training process in new environments, deployment of such systems could become faster and easier, enabling more widespread use and providing robust results.

Carreira and Zisserman [1] recently showed that there is an advantage to pre-training on large, related datasets for video-level action classification. However, no such investigation exists in the literature with regards to frame-based action classification. Some works have looked at Recurrent Neural Netowrks (RNNs) learning from multiple sources, but only with regards to synthetic and non-vision data [19].

In this paper we address these two issues by exploring the transfer performance of Long Short Term Memory recurrent networks (LSTMs) for frame-based action classification on three related datasets - Breakfast [10], 50 Salads [20] and MPII Cooking 2 [14].

2. Background

Two of the most commonly used benchmarks for action recognition are the HMDB51 [12] and UCF101 [17] datasets. These, along with more recent ones, such as ActivityNet [6] and Kinetics [1], are all based around video segment classification. That is, videos are pre-split by a human into segments containing only a single action (either by annotating action boundaries or by uploading a video tagged "long jump" or "fix bicycle"), and these segments are wholly classified. One of the best performing methods for this type of task is Temporal Segment Networks (TSN) [23], which extracts features from frames spaced evenly throughout the video clip. Features are extracted from two-stream CNNs based on the Inception architecture [21]. These features are classified and a consensus from all examined frames is used as the overall classification for the clip. Other well-known methods based on this two-stream RGB and optical flow approach which look at segments as a whole include [3], [9] and [16], amongst others. Alternatively, 3DCNNs have also been tested on video-level classification, though typically with slightly lower accuracies compared to two-stream CNNs [22] [8].

Whilst these methods are suitable for applications such as video tagging, they are not designed for on-the-fly frame-based action recognition, where each new video frame is classified as soon as it is first seen. This is because the architectures do include the concept of temporal memory - each clip is only examined in full and all frames are required for a consensus. In order to classify frames individually when they are first seen, it is helpful to include some

form of memory into a classification model. For example, this would be required if trying to determine the difference between "open fridge" and "close fridge," as a single frame would be ambiguous and thus insufficient in making the distinction.

One of the most widely-used methods for this, within a deep learning framework, is a Recurrent Neural Network (RNN), with the most common implementation being made up of LSTM. These have proven popular for video classification [13], where the last output of the LSTM is used as the classification label, but as labels are available at every frame they are well suited to frame-based classification.

Many works have looked at the transfer performance with regards to CNN training , and it is common practice for networks pre-trained on ImageNet [2] to be used as the starting point for many classification and recognition applications. Carreira and Zisserman [1] recently showed that improved results can be delivered on the HMDB51 and UCF101 datasets by pre-training on the large-scale Kinetics dataset. The classification model is first trained on the Kinetics dataset, followed by fine-tuning on the HMDB51 and UCF101 datasets separately, depending on which one is being evaluated.

There are, interestingly, fewer works considering transfer learning on RNNs. Spieckermann introduced Factored Tensor RNNs [19][18] to perform joint and transferred learning, but these were not tested on visual data. Rather, gas turbine emission time series were used, as well as a frictionless cart-pole simulation. No such investigation exists in the literature with regards to frame-based action classification.

Datasets with per-frame labels include Charades [15], Thumos 15 [7], Breakfast [10], 50 Salads [20], and MPII Cooking 2 [14]. As we will be investigating learning from multiple sources, it makes sense to concentrate on those which record similar tasks, but from different domains, so we use the Breakfast, 50 Salads and MPII cooking 2 datasets. Examples are given in Figure 1, and these datasets are covered in more detail in Section 3.

In this work, we wish to tackle the issue of transfer learning for frame-based action classification using RNNs. Our main contribution is an investigation into the transfer performance of an LSTM network fed by a deep learning architecture on three related datasets (Breakfast, 50 Salads, MPII Cooking 2). We show that consistent improvement in overall classification accuracy and training convergence can be obtained by pre-training the LSTM on related tasks.

3. Datasets

For this work, we will use three datasets, which are all based around activities in the kitchen. They thus share some action-level labels (*e.g.* add salt, pour oil) , but are captured in different rooms, with different viewpoints, participants and recipes. Intuitively there should be some form of information common to all tasks, which can be leveraged during a shared training process.

Table 1 gives a brief overview. For all datasets in this paper, 4 train/test splits are used, with 75% training and 25% testing data. All splits use leave-person-out, i.e. no participant appears in both training and testing sets from the same split.

3.1. Breakfast

The Breakfast dataset consists of 433 sequences performed by 52 participants, containing 3078 actions across 50 classes (including a background class) in 18 different kitchens. Multiple viewpoints are provided for each scene, and RGB video is provided in a 320x240 resolution at 15 FPS. All the actions are delivered as part of 10 breakfast routines such as "cooking scrambled eggs" and "making tea," which are performed in a freeform manner. For this work, we have chosen to use the lowest level action labels. Examples include "pour cereal" and "smear butter."

3.2. 50 Salads

The 50 Salads dataset consists of 25 participants each preparing 2 salads. We again use the lowest level action classes, of which there are 52 (including the background class which we have added). This gives a dataset size of 2967 labelled actions. RGB-D and accelerometer (attached to cooking implements) data is provided, but for this work just the RGB footage is used. This is 640x480 and recorded at 30 FPS. Example actions include "cut tomato prep," "cut tomato core," and "cut tomato post." These pre- and post-labels are not found in the other two datasets.

3.3. MPII Cooking 2

The MPII Cooking 2 dataset contains 275 sequences containing 14105 actions across 88 classes (including the background class which we have added). 30 participants were told to cook certain dishes, but were not given precise instructions so this dataset can be considered freeform. Footage is from one kitchen, and example actions are "shake," "spread," and "apply plaster." 30 FPS RGB video is provided at a resolution of 1624x1224.

To give an indication of how difficult action classification in these types of environments is, we first report the results obtained when using TSN on pre-segmented action clips from the Breakfast dataset, shown in Table 2. This suggests that, if the decision is made to proceed with an inception-based method, it is best to just consider the RGB stream. An advantage of just using the RGB stream (in addition to improved results) is that optical flow is expensive to compute. This is problematic for real-time frame-based

(a) Breakfast [10] (b) 50 Salads [20] (c) MPII Cooking 2 [14]

Figure 1: Example frames from the three datasets used in this work.

Dataset	Frames (x1000)	Sequences	Actions	Classes	Participants	Environments	Viewpoints
Breakfast [10]	3042	433	3078	50	52	18	3-5
50 Salads [20]	578	50	2967	52	25	1	1
MPII Cooking 2 [14]	2871	273	14105	88	30	1	1

Table 1: Overview of the three datasets used througout this paper. The 'Actions' column shows the number of times an action is annotated, and the 'Classes' column shows the number of distinct action classes.

Split	RGB stream	Flow stream	Combined
0	**18.76%**	7.22%	11.45%
1	**11.88%**	7.14%	8.90%
2	**19.87%**	7.65%	13.76%
3	**23.85%**	7.26%	15.18%
Avg.	**18.59%**	7.32%	12.32%

Table 2: TSN results when using RGB, optical flow, and two fused streams on the Breakfast dataset. Results are reported on 25 video segments, the default parameter in TSN

classification and, as such, it tends to be computed a priori offline.

We note that TSN reports segment classification accuracies of 68.5% and 94.0% on HMDB51 and UCF101 respectively, which is much higher than the 18.59% average obtained by the (best performing) RGB stream here. This could be due to clips of kitchen actions (*e.g.* pour milk and pour coffee) being more visually similar than actions used in HMDB51 and UCF101 (*e.g.* cartwheel and ride horse). In such cases, context is less informative in deciding the class label.

4. Method

This paper attempts to explore the potential of using related datasets (*i.e.* those recorded of similar tasks) to decrease the amount of training required for a new dataset. We argue that as training effort and/or time decreases, the potential of deploying assistive systems in new environments would increase.

Previous works have successfully shown that feature extraction in itself is dataset-specific, and requires fine-tuning for every new location. We will investigate whether the process of training an RNN, which sits above a CNN-based feature extraction stage and usually takes the last CNN layer as its input, can benefit from co-learning or pre-learning from related datasets. This argument stems from the fact that RNNs do not directly map observations to classification outputs, which are indeed viewpoint and location specific, but focus on the temporal dependencies of classification labels. Mathematically-speaking, assume x_t is the per-frame observation, and o_t is the per-frame classification result, typically encoded as a one-hot vector. A general RNN learns not only the contribution of the per-frame observations but also the contribution of previous frame(s) decisions,

$$o_t = \phi(W x_t + U o_{t-1}) \quad (1)$$

In 1, W shows the observation contributions while U is the output-to-output contribution over time. When training from related tasks, we argue that the memory model ϕ could benefit from pre-training on related tasks as opposed to random initialisations.

We first present how features can be extracted to produce dataset-specific single frame decisions (i.e. x_t). We then explain how these are fed into LSTMs - our choice of RNN architecture, and how cross-dataset training can be

Split	DT FV + SVM	CNN + Softmax	CNN + SVM
0	19.05%	22.00%	**24.77%**
1	**18.90%**	16.19%	17.01%
2	**22.15%**	19.90%	21.72%
3	20.53%	24.18%	**25.38%**
Avg.	20.16%	20.57%	**22.22%**

Table 3: Feature comparison on the Breakfast dataset. CNN + Sofmax refers to the RGB stream from the Inception network with the result taken as the most likely class. CNN + SVM refers to the same network, but with the softmax layer fed to an SVM for classification.

achieved.

4.1. Feature extraction

We first compare hand-crafted Fisher-Vector encoded Dense-Trajectories (provided by the authors of [11]) against the RGB inception architecture, as used in the spatial stream of TSN. For classification of the Fisher-Vectors, an RBF SVM is used, which is trained on 50000 frames (using more frames during training gives no improvement). A grid search with 5-fold cross-validation is used to find the best cost and gamma. Results are also reported with the RBF SVM applied to the softmax layer of the RGB inception stream (CNN+SVM), as well as just taking the most likely class from the softmax layer (CNN + Softmax).

Results are shown for all four splits in Table 3. On average, the output from the softmax layer of the RGB inception network outperforms the SVM trained on Dense-Trajectory Fisher-Vectors. A further improvement is found when using an SVM trained on the the output of the softmax layer. This confirms that the Inception architecture is well suited to this problem, and that there appears to be additional information within the softmax features for a recurrent network, in our case an LSTM, to learn from.

We carried out this preliminary study on the breakfast dataset as it is the most challenging dataset with multiple locations and viewpoints (see Table 1).

4.2. LSTM Architecture

To introduce an element of history into our model, we choose to use an LSTM. Greff et al. [5] evaluated 8 different LSTM cell types on speech recognition, handwriting recognition and polyphonic music modelling. They found that there was no significant improvement over the widely-used *vanilla* LSTM cell [4], which is what we will use in this work. The *vanilla* LSTM consists of input, output and forget gates, block input and output, and output activation function and peephole connections. An example of the effect an LSTM can have on frame-based CNN classifications is shown in Figure 2.

The softmax layer from the RGB Inception network is used as a feature, so the input dimensionality is the number of classes n in each dataset. Given a fixed amount of memory, there is a tradeoff between history size (*i.e.* how many times the network is unrolled) and depth (*i.e.* how many LSTM cells are stacked). We found that for these datasets it is preferable to prioritise a larger history over depth. The most likely explanation is that the CNN which feeds into the LSTM has already done most of the abstraction from visual data to labels. The problem being solved by the LSTM is therefore more straightforward, namely the temporal component. This suggests that long-term knowledge of which action has been performed earlier in the sequence is more valuable than better short-term insight.

To learn from long sequences, the training process involves randomly selecting a subsequence with a length equal to the LSTM history from a random video, and combining a set number of these into a batch. The LSTM loss function takes into account the loss from every frame, and is calculated as the average log loss across the sequence. This contrasts with snippet classification using LSTMs, where only the loss from the last frame is considered.

Following preliminary evaluations, all trained LSTMs in this paper share the following parameters; history: 300 frames, depth: 1, hidden layer size: 128, batch size: 128.

4.3. Joint and Transferred LSTM Training

A number of variations on the training process are trialled, in order to investigate whether any improvement can be found by training on multiple datasets. Traditionally, RNNs, including LSTMs, are trained from random initialisations. In this paper, we use three methods for cross-dataset learning including the baseline. These are:

1. None: Random initialisations for LSTM parameters
2. Joint: Training multiple datasets jointly
3. Transfer: Training on one dataset (referred to as the *source* dataset) then fine-tuning on the second (referred to as the *target* dataset).

To achieve Joint and Transfer learning, one first needs to tackle the issue of the difference in number and identity of classes in each dataset. Recall that for each LSTM, the dimensionality of the input vector x_t and the output vector o_t equals that of the number of classes n in the dataset. For one LSTM to be trained using multiple datasets, this dimensionality difference needs to be bridged.

We propose four different forms of label assignment. Given one dataset L with l labels and another dataset M with m labels ($l < m$), the first option is to map the labels from L onto the labels of M, starting at label 0, which, by

Figure 2: Example of a CNN and the same CNN with an LSTM on the Breakfast dataset (images are 5 frames apart). The CNN mistakes two "crack_egg" frames as "stir_dough" (the hand positions and pan seem to be in similar locations to a bowl when it is being stirred). The LSTM appears to have learned that stirring dough is unlikely to occur after cracking an egg, and can correct for this.

convention, is the "background" class. The dimensionality of the LSTM used in training is thus equal to the larger number of labels, i.e. m. This is called *merged label assignment*. By enforcing a link between one of the largest common classes, it is hoped that the LSTM can learn shared information in a better way. More explicitly, given a label s_i in any of the two datasets prior to label assignment, its assigned m dimensional form c_i post label-assignment is

$$c_i = \begin{cases} s_i & \text{if } s \in M \\ s_i & \text{if } s \in L \text{ and } i < l \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Random label assignment is similar to merged label assignment, but the mapping of L to the first l entries of M is done randomly, but still in a one-to-one manner. Here, the i'th element of the newly assigned m dimensional label c is

$$c_i = \begin{cases} s_i & \text{if } s \in M \\ s_{\text{rand}(i)} & \text{if } s \in L \text{ and } i < l \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

where $0 \leq \text{rand}(i) < l$ is unique for each i, *i.e.* is chosen without replacement. *Multiple random label assignments* are trialled as a form of data augmentation. In this paper, we using single (1) and multiple (3) random mappings per source dataset. The idea behind providing multiple random assignments is that the LSTM will learn long-term trends which are independent of labels.

Concatenated label assignment involves creating an $l + m$ dimensional label, with labels from L corresponding to the first l entries, and labels from M corresponding to the next m entries. For each dataset, the remaining labels are zero-filled for both the input and the output vectors. Here, the i'th element of the assigned $l + m$ dimensional label c is

$$c_i = \begin{cases} s_i & \text{if } s \in L \text{ and } i < l \\ s_{i+l} & \text{if } s \in M \text{ and } i \geq l \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

5. Experiments

We first present results from each split of all three datasets as a baseline for comparisons. The experimental pipeline is as follows:

1. Train Inception RGB stream on training portion of split.
2. Extract softmax layer responses for all images in the dataset.
3. Train LSTM using extracted softmax features of the training split.
4. Test LSTM using extracted softmax features of the test split.

Results for this process on all splits of all datasets are given in Table 4, which shows that the CNN+LSTM outperforms the CNN on its own in every case, as well as outperforming all features tested in Table 3. It also appears to make a greater contribution on the Breakfast and 50 Salads datasets compared to the MPII Cooking 2 dataset. This could be due to it containing more temporal label information to learn from - the Breakfast dataset contains a similar number of frames, but has fewer label sequences as these are duplicated in multiple viewpoints, and the 50 Salads dataset is roughly 5 times smaller.

5.1. Joint and Transferred Training Results

For Joint learning, batches made up of a ratio of 5:3 in favour of the dataset being evaluated gave the most promising results.

Table 5 shows the effect of different labels and Joint/Transfer learning on the worst performing split of the Breakfast dataset (split 1), jointly trained with or transferred from the best performing split from the 50 Salads dataset (also split 1). A baseline is given of the LSTM just trained on Breakfast split 1, as shown earlier in Table 4.

Table 5 suggests that multiple random label assignments can have a negative effect on classification accuracy, and

Dataset	Split	CNN	CNN + LSTM
Breakfast	0	22.00%	**30.78%**
Breakfast	1	16.19%	**19.53%**
Breakfast	2	19.90%	**28.75%**
Breakfast	3	24.18%	**32.26%**
50 Salads	0	35.00%	**37.86%**
50 Salads	1	40.41%	**48.77%**
50 Salads	2	34.43%	**38.22%**
50 Salads	3	40.07%	**45.84%**
MPII	0	31.73%	**36.84%**
MPII	1	35.97%	**39.47%**
MPII	2	39.56%	**40.74%**
MPII	3	37.13%	**39.71%**

Table 4: CNN frame-based classification performance compared against an LSTM with the CNN softmax layer as input.

Training	Label Arrangement	Accuracy
N/A	N/A	19.53%
Joint	Merged	20.78%
Joint	Concat	20.44%
Joint	Random	20.98%*
Joint	Multiple Random	19.02%*
Transfer	Merged	20.80%
Transfer	Concat	**21.37%**
Transfer	Random	20.06%*
Transfer	Multiple Random	19.69%*

Table 5: Evaluation of training methodologies and label arrangements on split 1 of the Breakfast dataset. Joint and transfer training is performed with split 1 of the 50 Salads dataset. A * indicates that an average is taken over three runs of random label maps. The first line is a benchmark with no transferred or joint learning.

that concatenated and merged label assignments can outperform random initialisation for transferred and joint learning on this split. However, the most promising result is provided by transferred training with concatenated labels.

We next carry out comprehensive evaluation of this proposal (i.e. Transfer training with concatenated labels) on the rest of the Breakfast dataset as well as all splits from the 50 Salads and MPII Cooking 2 datasets. Table 6 shows results on all splits of the Breakfast dataset. A small average improvement can be found when transferring from either the 50 Salads (0.68%) or MPII Cooking 2 (1.08%) datasets. Table 7 shows small performance improvements can be found when transferring from the Breakfast (1.30%) and MPII Cooking 2 (0.54%) datasets to the 50 Salads dataset. Table 8 completes this round of experiments, and

		Transfer Dataset	
Split	No Transfer	50 Salads 1	MPII 2
0	30.78%	32.23%	**34.01%**
1	19.53%	**21.37%**	20.56%
2	28.75%	28.51%	**28.77%**
3	32.26%	31.94%	**32.29%**
Avg.	27.83%	28.51%	**28.91%**

Table 6: Transfer performance of 50 Salads split 1 and MPII Cooking 2 split 2 with label concatenation on the Breakfast dataset.

		Transfer Dataset	
Split	No Transfer	Breakfast 1	MPII 2
0	37.86%	38.45%	**39.89%**
1	48.77%	49.07%	**49.31%**
2	38.22%	**40.61%**	38.11%
3	45.84%	**47.75%**	45.55%
Avg.	42.67%	**43.97%**	43.22%

Table 7: Transfer performance of Breakfast split 1 and MPII Cooking 2 split 2 with label concatenation on the 50 Salads dataset.

		Transfer Dataset	
Split	No Transfer	Breakfast 1	50 Salads 1
0	36.84%	**36.97%**	36.34%
1	39.47%	39.70%	**40.80%**
2	40.74%	41.33%	**41.64%**
3	39.71%	**39.85%**	39.12%
Avg.	39.19%	39.24%	**39.48%**

Table 8: Transfer performance of Breakfast split 1 and 50 Salads split 1 with label concatenation on the MPII Cooking 2 dataset.

shows smaller improvements when transferring from the Breakfast (0.05%) and 50 Salads (0.29%) datasets to the MPII Cooking 2 dataset.

5.2. Convergence

To investigate the effect of Transfer learning on LSTM convergence, one split is chosen from each of the three datasets. The fine-tuning process is performed pairwise on each combination of these three. First, training is run for 300,000 iterations on the source dataset with a learning rate of 0.001, then the network is fine tuned on the target dataset with the same learning rate, with a full test evaluation performed every 2^n iterations for $n = 9, ..., 14$. A comparison is given against a network trained on each target dataset, but

Figure 3: Convergence on split 1 of the Breakfast dataset when pre-trained with other datasets and random initial weights.

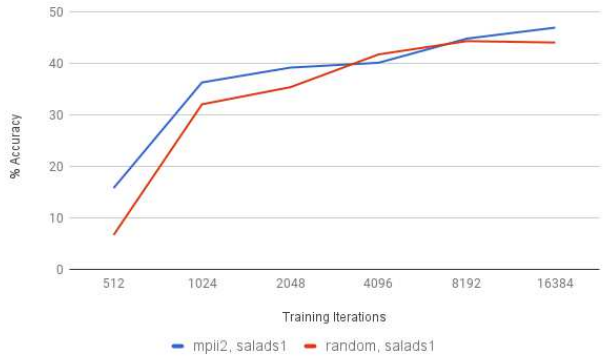

Figure 4: Convergence on split 1 of the 50 Salads dataset when pre-trained with other datasets and random initial weights.

with random initial weights.

The results for the convergence experiment are shown in Table 9. These are visualised in Figure 3 for the Breakfast dataset, Figure 4 for the 50 Salads dataset, and Figure 5 for the MPII Cooking 2 dataset. The Figures show how the random initial assignment baseline is outperformed by the best-performing transferred dataset. For all datasets, the LSTMs pre-trained on other datasets converge quicker, as shown in Table 9. Further, given a set number of iterations, the transferred training process gives better results than starting with random initial weights in the vast majority of cases. There is only one point (out of six where a full test is run) in each training sequence where one of the pre-trained LSTMs does not outperform random initialisation.

5.3. Discussion

The first point to note is that **Transfer learning is more successful than Joint training when considering only**

Figure 5: Convergence on split 2 of the MPII Cooking 2 dataset when pre-trained with other datasets and random initial weights.

LSTMs with these datasets. Transferring using label concatenation gives improved results over random initialisation in the experiments run on all three datasets, which suggests that shared temporal information is indeed being exploited. Another result we observe is that **transferring using label concatenation causes faster training convergence than random initialisation**, and that for a set number of iterations, classification performance is very likely to be higher.

Another interesting conclusion is that the MPII Cooking 2 dataset seems to transfer its temporal information to other datasets better - both in terms of overall performance and faster convergence, as well as benefiting from transferred information itself, although to a lesser extent. There are a number of possible reasons for this. One could be that it is a largest dataset containing more label sequences. Or it could be that its labelled classes are more general - examples include “clean,”, “close,” “chop” and “pour.” Labels from the 50 Salads dataset comparatively tend to be more specific - “add oil post,” “cut lettuce prep,” “mix salad core.” This point would be worthy of further investigation in future works.

6. Conclusion

In this work we have investigated whether frame-based action classification using deep neural networks and LSTMs can benefit from joint or transferred learning when only considering the LSTM component. We used three related datasets - Breakfast, 50 Salads and MPII Cooking 2, and discovered that small, reliable, improvements over the widely-used random initialisation could be found by pre-training LSTMs on datasets not being examined, and that label concatenation is the best way of handling this process. We also examined the convergence of LSTMs when pre-trained, and found that, as expected, they converged faster than random initial weights.

		Iterations					
Source	Target	512	1024	2048	4096	8192	16384
N/A	Breakfast 1	**6.42%**	10.20%	12.66%	14.85%	17.36%	18.08%
MPII 2	Breakfast 1	5.37%	**13.02%**	**16.04%**	**16.33%**	**18.00%**	**18.85%**
Salad 1	Breakfast 1	6.26%	10.94%	13.58%	14.91%	17.77%	18.13%
N/A	Salad 1	6.61%	31.97%	35.29%	**41.66%**	44.24%	43.97%
MPII 2	Salad 1	**15.72%**	**36.22%**	**39.10%**	40.04%	**44.73%**	**46.86%**
Breakfast 1	Salad 1	9.05%	32.92%	31.36%	41.39%	44.18%	46.82%
N/A	MPII 2	**27.08%**	28.59%	35.50%	37.30%	38.45%	39.36%
Breakfast 1	MPII 2	24.58%	32.73%	37.07%	**38.96%**	38.90%	**41.10%**
Salad 1	MPII 2	16.54%	**36.57%**	**38.52%**	37.71%	**40.19%**	39.63%

Table 9: Frame classification accuracy during the training process when using LSTM networks pre-trained on similar datasets. No source dataset indicates that random initial weights were used to initialise the LSTM.

This work presents a number of possible avenues for future research. These include attempting to understand why some datasets transfer better than others for training recurrent neural networks and whether other classification architectures are more well suited to, and can benefit more from, Transfer or Joint learning.

Data Statement & Ack: Public datasets were used in this work; no new data were created as part of this study. Supported by EPSRC LOCATE (EP/N033779/1).

References

[1] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Computer Vision and Pattern Recognition*, 2017. 1, 2

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition*, 2009. 2

[3] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *Computer Vision and Pattern Recognition*, 2016. 1

[4] A. Graves and J. Schmidhubera. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610, 2005. 4

[5] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, PP, 2017. 4

[6] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding. In *Computer Vision and Pattern Recognition*, 2015. 1

[7] H. Idrees, A. R. Zamir, Y. G. Jiang, A. Gorban, I. Laptev, R. Sukthankar, and M. Shah. The THUMOS challenge on action recognition for videos in the wild. *Computer Vision and Image Understanding*, 155:1–23, 2017. 2

[8] S. Ji, W. Xu, M. Yang, and K. Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. 1

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-Scale Video Classification with Convolutional Neural Networks. In *Computer Vision and Pattern Recognition*, 2014. 1

[10] H. Kuehne, A. Arslan, and T. Serre. The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities. *Computer Vision and Pattern Recognition*, 2014. 1, 2, 3

[11] H. Kuehne, J. Gall, and T. Serre. An end-to-end generative framework for video segmentation and recognition. In *Winter Conference on Applications of Computer Vision*, 2016. 4

[12] H. Kuehne, T. Serre, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: A large video database for human motion recognition. In *International Conference on Computer Vision*, 2011. 1

[13] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond Short Snippets: Deep Networks for Video Classification. In *Computer Vision and Pattern Recognition*, 2015. 2

[14] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A Database for Fine Grained Activity Detection of Cooking Activities. *Computer Vision and Pattern Recognition*, 2012. 1, 2, 3

[15] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding. In *European Conference on Computer Vision*, 2016. 2

[16] K. Simonyan and A. Zisserman. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances In Neural Information Processing Systems*, 2014. 1

[17] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *arXiv 1212.0402*, 2012. 1

[18] S. Spieckermann and D. Siegmund. Exploiting Similarity in System Identification. In *European Symposium on Artificial Neural Networks*, 2014. 2

[19] S. Spieckermann, S. Udluft, and T. Runkler. Data-effiicient temporal regression with multitask recurrent neural networks. In *Advances In Neural Information Processing Systems Workshop on Transfer and Multi-Task Learning*, 2014. 1, 2

[20] S. Stein and S. J. McKenna. Combining Embedded Accelerometers with Computer Vision for Recognizing Food Preparation Activities. In *International Joint Conference on Pervasive and Ubiquitous Computing*, 2013. 1, 2, 3

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *Computer Vision and Pattern Recognition*, 2016. 1

[22] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *International Conference on Computer Vision*, 2015. 1

[23] 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 *European Conference on Computer Vision*, 2016. 1