Neural Task Success Classifiers for Robotic Manipulation from Few Real Demonstrations

Abdalkarim Mohtasib
School of Computer Science
University of Lincoln
Lincoln, UK
amohtasib@lincoln.ac.uk

Amir Ghalamzan E.
Lincoln Institute for Agri-Food Technology
University of Lincoln
Lincoln, UK

Nicola Bellotto
School of Computer Science
University of Lincoln
Lincoln, UK

Heriberto Cuayáhuitl
School of Computer Science
University of Lincoln
Lincoln, UK

***Abstract*—Robots learning a new manipulation task from a small amount of demonstrations are increasingly demanded in different workspaces. A classifier model assessing the quality of actions can predict the successful completion of a task, which can be used by intelligent agents for action-selection. This paper presents a novel classifier that learns to classify task completion only from a few demonstrations. We carry out a comprehensive comparison of different neural classifiers, e.g. fully connected-based, fully convolutional-based, sequence2sequence-based, and domain adaptation-based classification. We also present a new dataset including five robot manipulation tasks, which is publicly available. We compared the performances of our novel classifier and the existing models using our dataset and the MIME dataset. The results suggest domain adaptation and timing-based features improve success prediction. Our novel model, i.e. fully convolutional neural network with domain adaptation and timing features, achieves an average classification accuracy of 97.3% and 95.5% across tasks in both datasets whereas state-of-the-art classifiers without domain adaptation and timing-features only achieve 82.4% and 90.3%, respectively.**

***Index Terms*—Deep Learning, Reward Learning, Task Success, Task Timing, Domain Adaptation, Robot Skill Learning**

I. INTRODUCTION

Large amounts of tasks are carried out in our daily lives, in large variation either due to the way they are done or to the environments where they are executed, and robots are expected to learn some of these tasks to be able to assist humans. In order for this to happen, robots should be able to learn tasks in an autonomous fashion as opposed to hard-coding all robot skills. Humans are able to learn new skills rather rapidly and, in many cases (not always because some tasks are difficult to master), using only a few examples. Arguably, robots should be endowed with similar or even better learning abilities to be able to acquire new tasks quickly and efficiently. Being able to identify the task goal and to measure task success is the first key aspect for robots to acquire new tasks autonomously.

Reward functions in robot learning play a major role in measuring task success in order to numerically reward the behaviour of robots [1]. The use of onboard robot sensors only—without relying on any other external sensors—makes the problem of measuring the success of tasks even harder. In this context, this paper focuses on training success classifiers (also referred to as 'goal classifiers') for measuring the levels of success in robotic manipulation tasks from only a few (as opposed to many) human demonstrations. The idea of training a success classifier in a new task using only a few demonstrations with high accuracy is still a challenging research problem. In this work, we study such a problem via the scenario illustrated in Fig. 1.

Our main contribution in this paper is a comprehensive comparison of different neural architectures for task success classification, based on feedforward neural networks, fully convolutional neural networks, sequence2sequence classifiers, domain adaptation, and a novel combination of them. This study was carried out using a newly proposed dataset of human-robot demonstrations in the Kitchen domain as well as an existing dataset of demonstrations [2] using a variety of manipulation tasks including stacking, placing, opening, closing, rolling, pushing, pulling, and rotating objects.

II. RELATED WORK

The topic of reward learning for trainable robots has been studied in several ways in the academic literature. [3]–[6] have studied how to solve this problem using Active Learning, which involves querying the expert user for labelling some trajectories or environment states. Similar approaches have combined the use of active learning together with Inverse Reinforcement Learning (IRL) [3], [4], where IRL is used to estimate and optimise a reward function by learning from demonstration in Markov Decision Processes (MDPs) [7]. Meta-Learning has also been used by robots that learn to acquire new tasks via knowledge transfer from a large set of pre-learned tasks [8], [9] to a new task. While meta-learning approaches have achieved promising results in learning the goals of new tasks from few demonstrations, they unfortunately require a very large number of training examples to train the meta-learner. Furthermore, the 'goal classifier' approach has been adopted by different researchers [10]–[12]. It uses an image-based classifier to predict whether an environment state (an image of the task at time t) represents a success or non-success of the executed task. Those goal classifiers usually require a substantial number of training examples, as well as an extensive effort from human demonstrators to be able

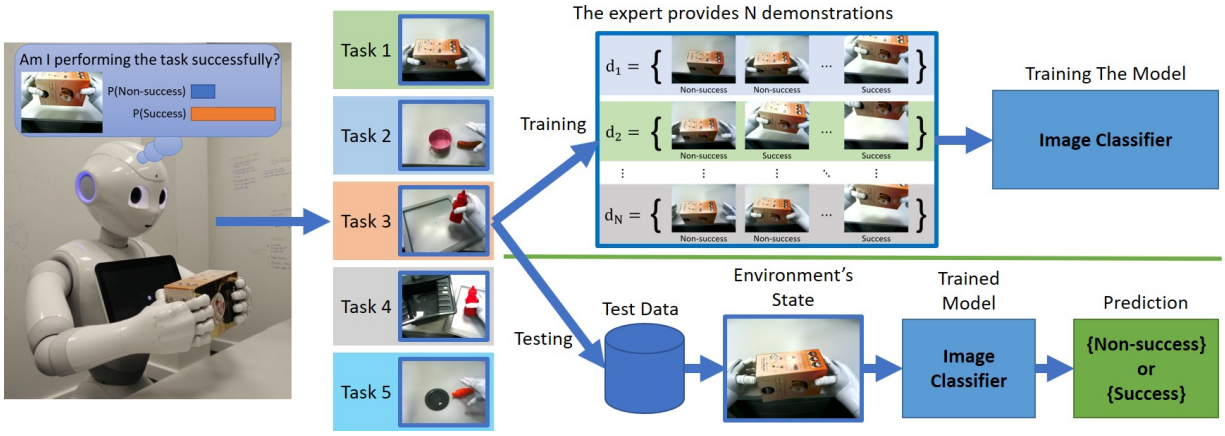

Fig. 1: Illustration of the targeted robot learning scenario

to successfully train task success predictors—this means that their effective and efficient training must be studied further.

Previous works are limited by relying on engineered reward functions [13], [14] and by using additional sensors (i.e. in addition to the onboard sensors) to estimate task success [15]–[17]. Other previous works have investigated the idea of learning the reward function from demonstration data [3], [18]–[25] mainly by using Inverse Reinforcement Learning—but they are unfortunately very data intensive. Other research projects have used active learning to query the expert for labelling uncertain states [10] or execution trajectories [5] to learn the task rewards—but their practical deployment to end users is unclear. Our work differs from previous ones in automatically inducing reward functions from raw data, in studying data-efficient methods for their practical application, and in the use of onboard sensors only (a 2D camera in our case) without any external sensors in the environment.

The main related works to ours that have used task success classifiers are mostly based on Convolutional Neural Networks (CNNs) [8], [10]–[12], [26]–[30], but they have not exploited sequential and timing aspects of robotic manipulation tasks for inducing their decisions. This paper investigates the effects of using timing information in manipulation tasks and sequential behaviour in an attempt to improve the performance of task success classifiers. In addition, domain adaptation techniques [8], [10]–[12], [26]–[30] have not been applied to reward learning. Domain adaptation refers to the case where what has been learnt in one domain (demonstrations in our case) is exploited to improve generalisation in another domain (unseen conditions in our case) [31]. This paper investigates the effects of using domain adaptation techniques to improve the performance of reward predictors casted as success classifiers. Thus, the context of this paper is to develop trainable reward models (as opposed to hard-coded) that can be used to accurately measure task success in robotic manipulation tasks. Future works can use such models as a part of numerical rewards required for robot learning systems. The code, models, and data produced as part of this paper are publicly available on GitHub[1].

III. Research Methods

A. Problem Definition

We consider a success classifier $g = f(s)$, where s is the environment state (an image or sequence of images from the robot's onboard 2D camera), and $g \in [0, 1]$ is the probability of having achieved the task in state s. This classifier can be used as a reward function for robot learning systems. The aim is to train $f(s)$ for a new manipulation task M_n from N demonstrations by updating the parameters of g to minimize $\sum \mathcal{L}(f(s_i), y_i)$, where $\mathcal{L}$ is the classification loss (cross entropy loss and mean square error in our case). We define $D_n = \{d_1, d_2, \cdots, d_N\}$ as the demonstrations dataset for the new task M_n, and each demonstration is defined as a set of states s and their labels y as follows: $d_i = \{(s_1, y_1), (s_2, y_2), \cdots, (s_j, y_j)\}_i$. The label y is 0 if the state s represents a Non-success in the task being executed, and 1 if the state s represents a Success in the task being executed. The research question that our study aims to answer is: *Can a task success classifier be trained effectively from a very small number of demonstrations (e.g. five)?*

B. Datasets

Our proposed dataset of human-robot demonstrations was collected using the Pepper Robot[2]. In this dataset, a human demonstrator performed kinesthetically the manipulation tasks for the robot by grabbing the robot hands and performing tasks in the Kitchen domain. During the task demonstrations, color images from the robot's 2D camera (size $320 \times 240 \times 3$) were recorded. When a demonstration task is completed, the demonstrator touched a tactile sensor on the robot head to signal task success. In this way and for practical purposes, any data collected before the tactile sensor was touched is labelled as *Non-success*, and any data collected after that is labelled as *Success*. The robot collects data at a sampling rate

[1]https://Mohtasib.github.io/RewardLearning/
[2]https://www.softbankrobotics.com/emea/en/pepper

TABLE I: Example training and test images in the Kitchen and MIME datasets, K_i and M_j, respectively

Dataset	Task	Training Examples	Test Examples
Kitchen	K_1		
	K_2		
	K_3		
	K_4		
	K_5		
MIME	M_1		
	M_2		
	M_3		
	M_4		
	M_5		
	M_6		
	M_7		
	M_8		
	M_9		
	M_{10}		

TABLE II: Training and test examples in the Kitchen dataset, per task, containing Non-Success (NS) and Success (S) images

#	Description	Training NS / S	Test NS / S
K_1	Grasp & lift a box	1281 / 194	74 / 85
K_2	Pick up a sausage & place it in a cooker	946 / 148	138 / 86
K_3	Pick up a ketchup & place it on a tray	1533 / 199	232 / 67
K_4	Pick up a ketchup & place it in a sink	1093 / 264	162 / 123
K_5	Pick up a carrot & place it on a plate	1338 / 257	311 / 69

TABLE III: Summary of tasks in the MIME dataset

#	Description	Training	Test
M_1	Stack	888 / 103	828 / 96
M_2	Place objects in box	1718 / 360	1576 / 283
M_3	Open bottles	1755 / 133	1594 / 135
M_4	Push (Single hand)	323 / 60	301 / 80
M_5	Rotate	808 / 135	778 / 130
M_6	Close Book	604 / 177	555 / 168
M_7	Pull (Two hands)	1538 / 145	1515 / 151
M_8	Push (Two hands)	1816 / 384	1879 / 333
M_9	Roll	489 / 100	377 / 130
M_{10}	Pull (Single hand)	544 / 94	466 / 83

of 10 samples/sec. Example images captured using the Pepper robot's 2D camera for grasping, lifting, picking up and placing kitchen objects are illustrated in Fig. 1.

While demonstration data of five different robotic manipulation tasks were collected (see list of tasks in Table II), each of these demonstrations was carried out at a different speed and with objects positions initialised randomly. The randomisation

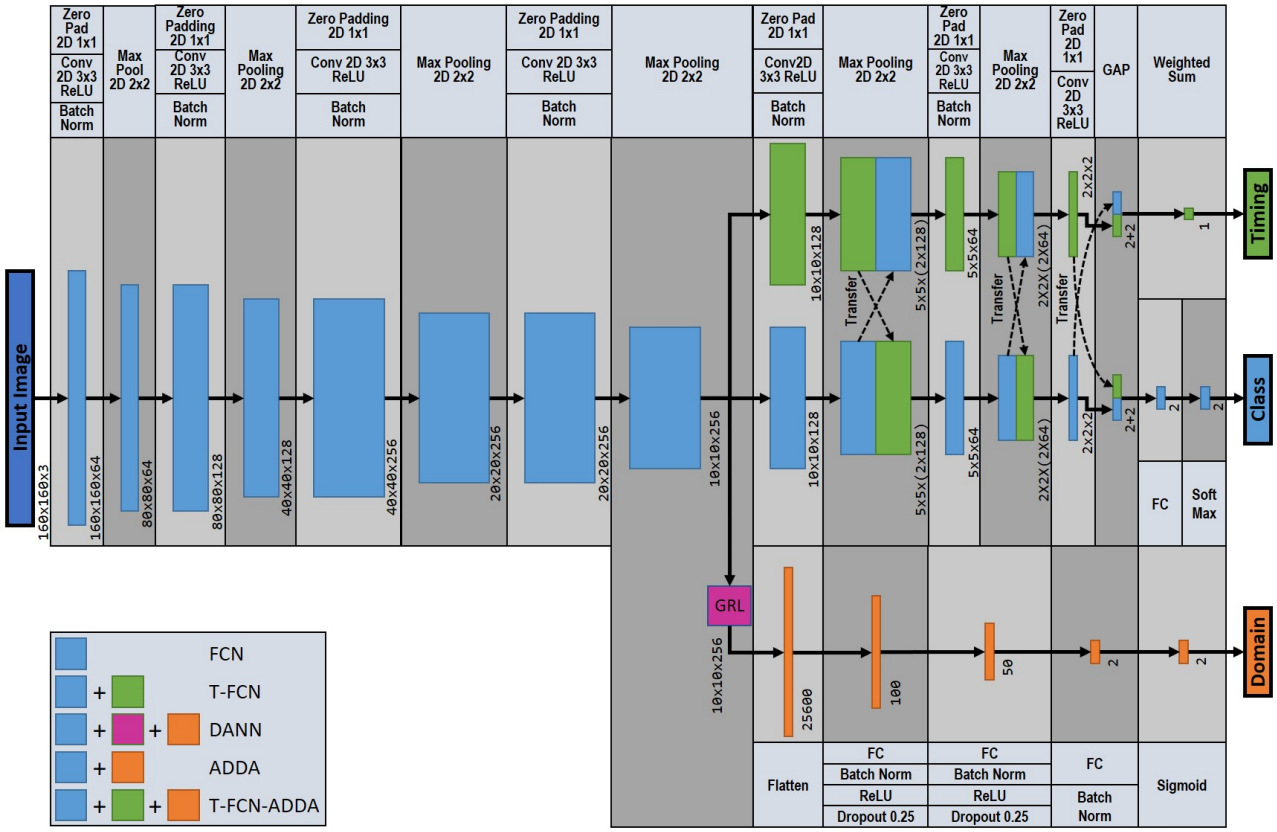

Fig. 2: CNN-based neural architectures for task success classification

was limited to the workspace of the robot's arms and to the view of the robot's camera (see K_i examples in Table I).

The MIME (Multiple Interactions Made Easy) dataset has been used for learning from demonstrations in multiple tasks [2]. It consists of kinesthetic trajectories and videos of human demonstrations collected using the Baxter robot (https://en.wikipedia.org/wiki/Baxter_(robot)), i.e. human demonstrations and their corresponding robot demonstrations. To collect this dataset, a set of human demonstrators were trained to handle the robot before performing the demonstration task. Every human demonstrator provided multiple demonstrations for each task using different objects –filtering out incorrect cases. The dataset contains in total 8260 demonstrations for 20 tasks, from which we extracted 10 demonstrations (5 for training and 5 for testing) for the 10 M_j tasks shown in Table I. These 10 tasks and their demonstrations have been selected randomly.

C. Model Architectures

The following neural architectures are studied in this paper.

- *Fully Connected Neural Net* (**NASNet**): This baseline is NASNet-based [32] with pre-trained weights on ImageNet [33]. It has the best classification performance compared to the other models implemented in the Keras API [34]. In our model, features extracted from NASNet are passed to a Feed-Forward Network consisting of six fully-connected layers as illustrated in Fig. 3.

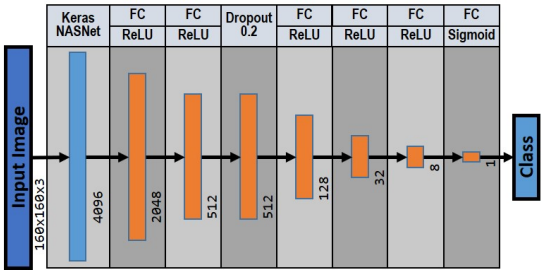

Fig. 3: NASNet-based neural architecture

- *Fully Convolutional Neural Net* (**FCN**): This second baseline is very similar to the CNN-based models that have been used in literature as success classifiers [8], [10], [11], [26]–[30]. This network is the core of other architectures implemented in this paper. The inputs to FCN are (160 × 160 × 3) resized images from the robot's 2D camera, followed by six main convolutional blocks and one convolutional layer, see Fig. 2.
- *Time-Based Fully Convolutional Neural Net* (**T-FCN**): This architecture extends FCN with two paths and features (shared in between): one is the classification path, the other is a timing path that predicts the proportion of task completion (a regressor), see Fig. 2. The task completion proportion for each image is calculated according

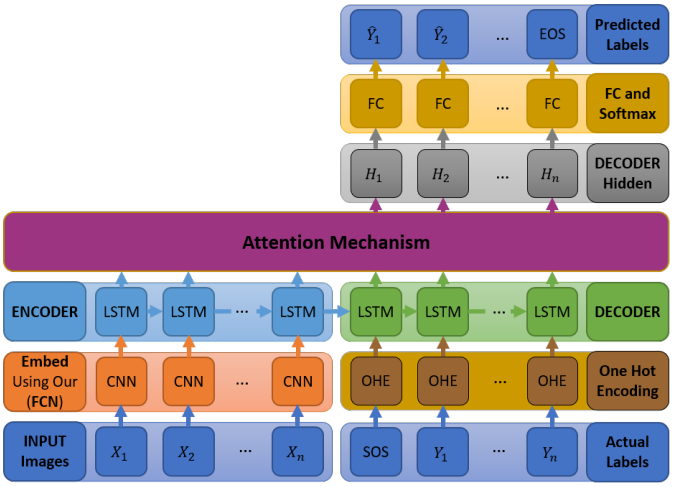

Fig. 4: Encoder-Decoder neural architecture

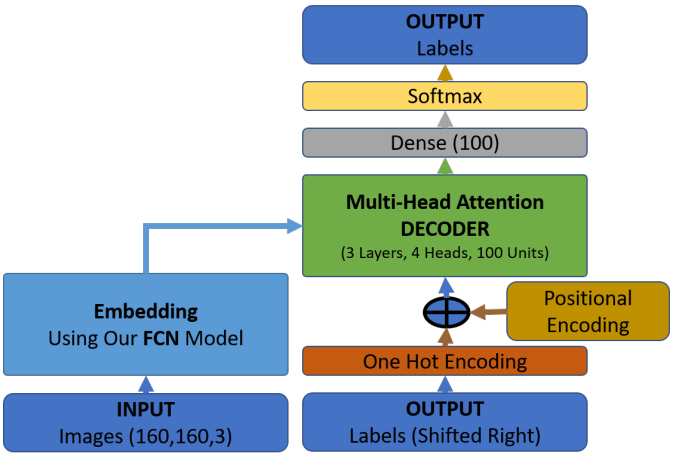

Fig. 5: Transformer neural architecture

to $y_T = \frac{t}{(j-1)}$, where t is a given time step and j is the total number of time steps in the demonstration at hand. This neural architecture is novel as the timing features have never been used in this manner before. We are not interested in predicting the task completion proportion, but this branch will help to learn more features about the task during the model training.

- *Attention-Based Encoder-Decoder* (**Attention-RNN**): This architecture is an attention-based encoder-decoder using LSTM-based recurrent neural nets with Bahdanau attention [35]. While the encoder neural net generates features out of a history of images (10 in our case), the decoder predicts the sequence of labels. The inputs to the encoder are features extracted using the FCN model from input images. The inputs to the decoder are one hot encodings of the classification labels (see Fig. 4).
- *Transformer Network* (**Transformer**): This architecture [36] has achieved state-of-the-art results in a number of applications, especially in natural language processing [37]–[39]. Similarly to **Attention-RNN**, it uses the **FCN** model for embedding input images and it also uses a history of 10 images. The decoder predicts the sequence of labels based on the multi-head attention layers and the features produced by our **FCN** model (see Fig. 5).
- *Domain-Adversarial Neural Network* (**DANN**): This architecture extends our **FCN** network by adding domain adaptation using the so-called Domain-Adversarial Training of Neural Networks (DANN) [40], [41]. The main component of its domain discriminator path is a Gradient Reversal Layer (GRL), which reverses the gradient sign during backpropagation (see Fig. 2).
- *Adversarial-Discriminative Domain Adaptation* (**ADDA**): This is similar to **DANN**, the only difference is in the domain adaptation method. Here we used Adversarial Discriminative Domain Adaptation (ADDA) [42], which uses adversarial weights instead of GRL.
- *Timing-Based and Domain-Based Fully Convolutional Net* (**T-FCN-ADDA**): This architecture is similar to our **T-FCN** architecture, but it uses three paths instead of two: classification, timing, and domain. It is a novel neural architecture that combines the **T-FCN** and **ADDA** architectures above (see Fig. 2). Similar to the T-FCN model, we are only interested in predicting the success probability using the classification path, but the timing and domain paths will help to learn more features about the task in hand.

IV. Evaluation

A. Experimental Setting

For each manipulation task, we split the demonstration data (images and classification labels) into a training set (80%) and a validation set (20%), while the unseen conditions data is used as a test set. In terms of loss functions, we used cross-entropy as a loss function for the success classifiers and the domain discriminator, and the mean square error for the timing predictor[3]. This is due to the fact that the success predictors and domain discriminators are classification tasks, and the timing predictor is a regression task. All of the following experiments and tests were carried out for all fifteen tasks listed in Tables I, II, and III. Our experiments[4] focus on assessing the performance of success classification according to the following metrics: Classification Accuracy, Precision, Recall, F1-score, Area Under the Curve (AUC), Training Time, and Test Time. The latter two metrics refer to average times (in seconds) per image.

B. Experimental Results: Overview

The performance of our architectures—shown in Table IV—is analysed in three groups. First, we compare **NASNet** vs.

[3]Summary of hyperparameters: batch size=16, epochs=100, optimizers=adam, and learning rate=0.001.

[4]PC specs: **CPU**: Intel i7-4770 @ 3.40GHz. **RAM**: 16GB. **GPU**: NVIDIA GeForce GTX 750 Ti 2GB.

TABLE IV: Average performance results of our baseline and proposed neural architectures for task success classification applied to the Kitchen and MIME datasets (notation: **ACC**=Average Classification Accuracy, **AUC**=Area Under the Curve)

Architecture	Training Time	Test Time	Kitchen Dataset					MIME Dataset				
			ACC	Precision	Recall	F1 Score	AUC	ACC	Precision	Recall	F1 Score	AUC
NASNet	0.8044	0.8013	0.6190	0.1670	0.1612	0.1634	0.4192	0.8809	0.6186	0.6627	0.6165	0.7173
FCN	0.1373	0.0283	0.8240	0.9128	0.5244	0.6268	0.7586	0.9032	0.6211	0.8277	0.6917	0.7981
T-FCN	0.1921	0.0564	0.9131	0.9194	0.7716	0.8058	0.8636	0.8683	0.7694	**0.8612**	0.7660	0.8346
Attention-RNN	0.2133	0.0724	0.8380	0.8642	0.6570	0.6948	0.8776	0.8555	0.6870	0.6800	0.6278	0.7701
Transformer	0.2102	0.0681	0.8570	0.9130	0.5694	0.6338	0.7796	0.8576	0.6845	0.7443	0.6453	0.8084
DANN	1.9470	0.3250	0.9176	0.9052	0.7954	0.8334	0.8914	0.9410	0.8893	0.7438	0.7841	0.8372
ADDA	0.2082	0.0455	0.9577	0.9202	0.8980	0.9152	0.9300	0.9409	0.8668	0.7490	0.7612	0.8307
T-FCN-ADDA	0.2209	0.0564	**0.9733**	**0.9950**	**0.9052**	**0.9452**	**0.9642**	**0.9552**	**0.9429**	0.8042	**0.8397**	**0.9070**

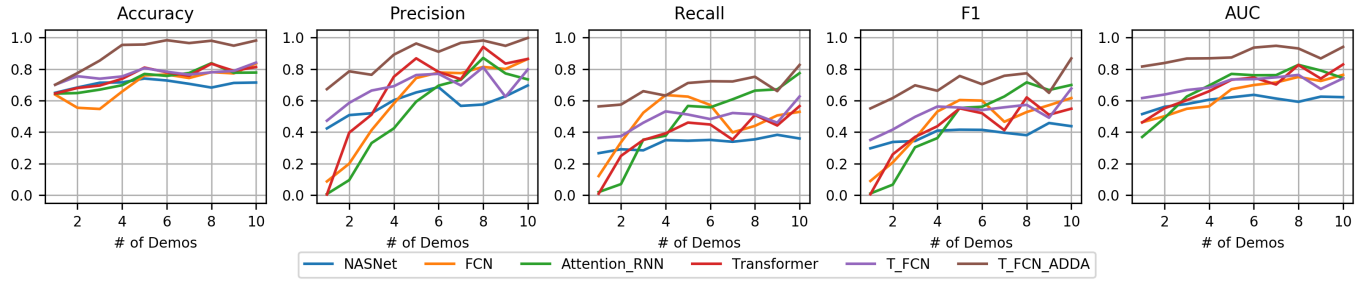

Fig. 6: Performance results for different amounts of demonstrations, from 1 to 10

FCN to observe their performance when trained using the data of five demonstrations and tested on unseen conditions with unseen distractor objects and different backgrounds. We also compare vanilla **FCN** vs. **T-FCN** to study the effects of using timing features. Second, we compare **Attention-RNN** vs. **Transformer** to study the sequential aspect of the manipulation tasks, and contrast their performance against the previous group. Third, we investigate and compare two domain adaptation methods (**DANN** and **ADDA**) for mapping from the source domain (training) to the target domain (test=unseen conditions). In addition, we study the performance of **T-FCN-ADDA**—a combined model that uses both the timing and domain adaptation aspects on top of FCN learnt representations.

a) Fully-Connected and Fully-Convolutional Classifiers: Table IV shows that **T-FCN** outperforms **NASNet** and **FCN** across most of the metrics. The difference in performance according to F1-Score is more stark than the other metrics (ACC and AUC). It can also be noted that **NASNet** is computationally more expensive, not only during training but also at test time. Assuming the classification of one million images, while **FCN** would require 7.9 hours, **T-FCN** would require 15.7 hours, and **NASNet** would require 225.6 hours. Thus, **T-FCN** is not only the most accurate in this group, but it is also substantially faster at training and prediction times than **NASNet**—even when it is a slower predictor than **FCN**.

b) Seq-to-Seq Classifiers: Table IV shows that **Transformer** outperforms **Attention-RNN** according to ACC, but not according to F1-Score and AUC. Given that our dataset is imbalanced, we could rely on F1-Scores instead of ACC; but we found that we can rely on ACC. We found strong correlations between ACC and F1, ACC and AUC, and F1 and AUC, obtaining Pearson correlation coefficients of 0.79, 0.64, and 0.91, respectively. The fact that both classifiers in this group are outperformed by the best classifier of the first group (**T-FCN**) (across all metrics), suggests that further research on this type of architectures is needed to integrate sequential aspects into the top success classifier.

c) Domain Adaptation Classifiers: Table IV shows that the use of domain adaptation techniques (**DANN** and **ADDA**) helps to achieve better results than the architectures in the previous groups. Although **ADDA** outperforms **DANN** across most metrics, the combination of **T-FCN** and **ADDA** (i.e. **T-FCN-ADDA**) achieves the best classification results. While **T-FCN** and **ADDA** achieve average F1-scores of (80.6%, 76.6%) and (91.5%, 76.1%) across tasks for both datasets (respectively), **T-FCN-ADDA** achieves an average F1-scores of (94.5%, 84%) in both datasets. Regarding prediction times, **T-FCN-ADDA** is comparable to **T-FCN**—the third fastest in our neural architectures, just after vanilla **FCN** and **ADDA**. This is due to the fact that the domain discriminator is not used at test time.

C. Experimental Results: Analysis

A natural question to ask is "How many demonstrations are needed to train task success classifiers?'. We analysed the answer to this question and found that there is a small improvement in performance when increasing the number of demos used for training from 5 to 10 demos (see Fig. 6). This suggests that the slightly small improvement in performance is not worth the double efforts from human demonstrators. This result justifies our selection regarding the use of 5 demonstrations to train our models.

Task	FCN		T-FCN		ADDA		T-FCN-ADDA	
	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
κ_1	0.903	0.823	0.892	0.906	0.971	0.933	0.911	0.975
κ_2	0.817	0.933	1.000	0.835	1.000	0.945	1.000	1.000
κ_3	0.832	0.808	0.840	0.800	0.927	0.907	0.967	1.000
κ_4	0.633	1.000	0.988	0.992	1.000	0.984	1.000	1.000
κ_5	0.899	1.000	0.907	1.000	0.951	0.946	0.942	1.000
μ_1	1.000	0.345	0.934	0.905	0.908	0.684	0.956	0.967
μ_2	0.999	0.940	0.991	0.985	0.997	0.969	0.984	1.000
μ_3	0.999	0.609	0.996	0.977	1.000	0.985	0.999	0.985
μ_4	1.000	0.825	1.000	0.571	1.000	0.800	1.000	0.870
μ_5	0.998	0.508	0.938	0.599	0.999	0.956	0.966	0.912
μ_6	0.938	1.000	0.898	1.000	0.967	0.829	0.987	0.964
μ_7	0.995	0.593	1.000	0.645	0.995	0.911	0.996	0.960
μ_8	0.978	0.809	0.995	0.861	0.996	0.985	0.998	0.988
μ_9	0.744	0.000	0.000	0.256	0.779	0.889	0.781	1.000
μ_{10}	0.979	0.357	0.968	0.490	0.896	0.667	0.940	0.783
Avg.	0.914	0.703	0.890	0.788	0.959	0.893	**0.962**	**0.960**
Std.	0.112	0.296	0.251	0.226	0.061	0.104	**0.057**	**0.062**

TABLE V: Performance of our **T-FCN-ADDA** classifier and three baselines. Notation: TPR=TP Rate, TNR=FP Rate, TPR=TP/(TP+FN), TNR=TN/(TN+FP), TP=True Positives, TN=True Negatives, FP=False Positives, FN=False Negatives

From Table V it can be noted that the true positive rates (TPR, or recall) and false positive rates (TNR) are higher for **T-FCN-ADDA** than its counterparts. This is clearly the case on average for all tasks in both datasets, but also the case on most individual tasks. These results show evidence that our top classifier (proposed) is more reliable than the baselines.

To illustrate the performance of our task success classifiers, Figs. 7 and 8 show the predicted success probabilities for example demonstrations of our top classifier (**T-FCN-ADDA**) and three baseline classifiers (**FCN**, **T-FCN**, and **ADDA**). A visual inspection shows that our top classifier is closer to the ground truth than the baselines in both datasets.

V. Concluding Remarks

This paper studies the problem of reward learning for robotic manipulation via neural task success classifiers, where the aim is to find out whether it is possible to train accurate task success classifiers from few demonstrations. Our study is a step in the direction of autonomous robot skill acquisition, where a human demonstrator shows a humanoid robot how to carry out a novel task. Our experiments are focused on predicting (probabilistically) whether the robot has achieved its task or not. We carry out a comprehensive comparison of three types of neural architectures: (i) fully-connected and fully-convolutional neural nets, (ii) sequence-to-sequence learning, and (iii) domain adaptation learning.

Our experiments use two datasets containing images on different tasks and with varied backgrounds and distractor objects. Experimental results reveal that 5 human demonstrations is a good compromise between effort and performance. They also show that **T-FCN** is the best architecture from the first group; **Transformer** is the best from the second group according to classification accuracy; **Attention-RNN** is the best according to F1-score; and **T-FCN-ADDA** is the best architecture not only from the third group but from the three groups of classifiers. **T-FCN-ADDA** is a novel solution that

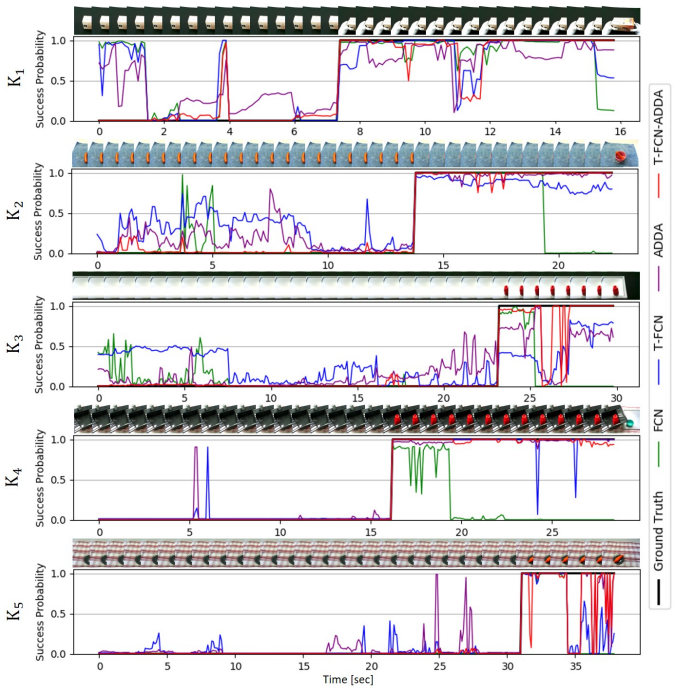

Fig. 7: Illustration of predicted probabilities generated by **T-FCN-ADDA** and three baselines on the Kitchen dataset.

combines the best architecture from group one and the best domain adaptation classifier. It achieves high performance across all tasks in the Kitchen and MIME datasets with average classification accuracies of **97.3%** and **95.5%** in those datasets, while vanilla **FCN** models only reach **82.4%** and **90.3%**, respectively.

Future works include investigating reward learning in more complex tasks than those attempted here, training robots to carry out manipulation tasks using the proposed classifiers, and studying their application to other robot platforms.

References

[1] Jens Kober, J. Andrew Bagnell, and Jan Peters, "Reinforcement learning in robotics: A survey," *I. J. Robotics Res.*, vol. 32, no. 11, 2013.

[2] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta, "Multiple interactions made easy (MIME): large scale demonstrations data for imitation," in *CoRL*. 2018, vol. 87, PMLR.

[3] Manuel Lopes, Francisco Melo, and Luis Montesano, "Active learning for reward estimation in inverse reinforcement learning," in *ECML-KDD*, 2009.

[4] Yuchen Cui and Scott Niekum, "Active reward learning from critiques," in *ICRA*, 2018.

[5] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters, "Active reward learning.," in *RSS*, 2014.

[6] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia, "Active preference-based learning of reward functions.," in *RSS*, 2017.

[7] Andrew Y Ng, Stuart J Russell, et al., "Algorithms for inverse reinforcement learning.," in *ICML*, 2000, vol. 1.

[8] Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn, "Few-shot goal inference for visuomotor learning and planning," in *CoRL*, 2018.

[9] Stephen James, Michael Bloesch, and Andrew J. Davison, "Task-embedded control networks for few-shot imitation learning," in *CoRL*, 2018, vol. 87.

[10] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine, "End-to-end robotic reinforcement learning without reward engineering," in *RSS*, 2019.

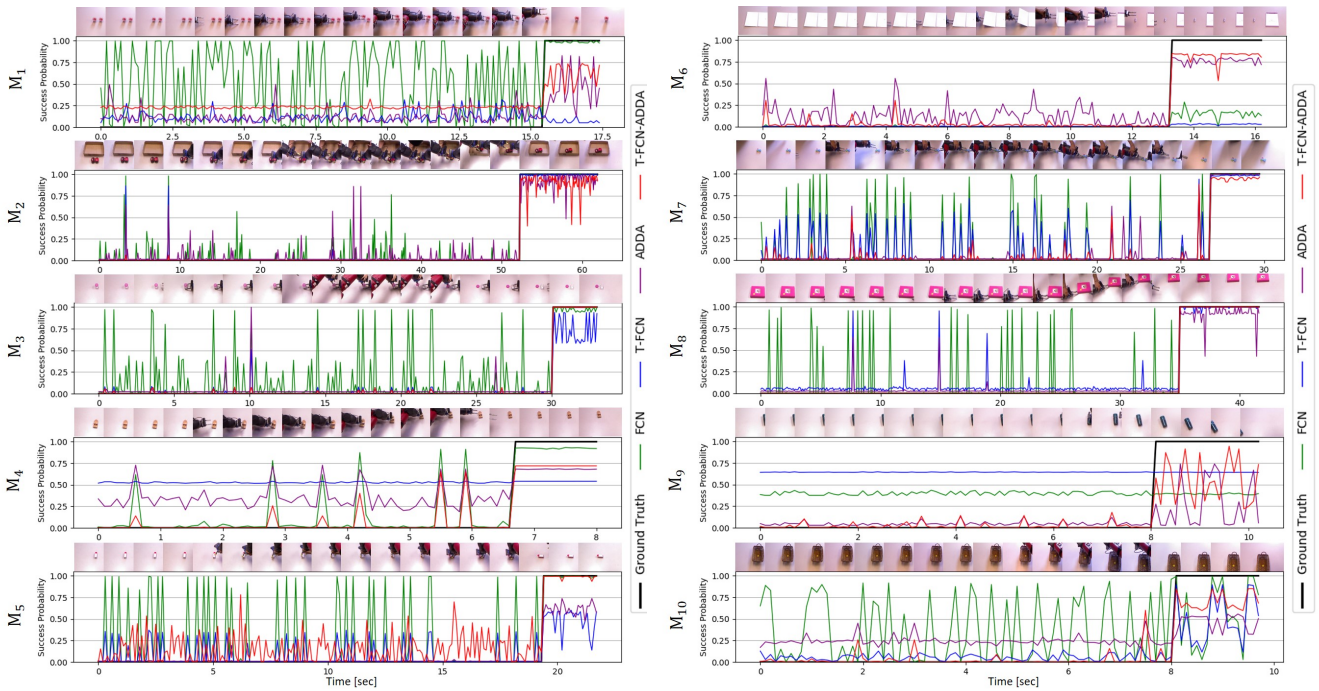

Fig. 8: Illustration of predicted probabilities generated by **T-FCN-ADDA** and three baselines on the MIME dataset.

[11] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz, "A practical approach to insertion with variable socket position using deep reinforcement learning," in *ICRA*, 2019.

[12] Lerrel Pinto and Abhinav Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *ICRA*, 2016.

[13] Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *CoRR*, 2017.

[14] Stephen James and Edward Johns, "3d simulation for robot arm control with deep q-learning," *CoRR*, 2016.

[15] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in *IROS*, 2017.

[16] Akihiko Yamaguchi, Christopher G Atkeson, and Tsukasa Ogasawara, "Pouring skills with planning and learning modeled from human demonstrations," *Intl. Journal of Humanoid Robotics*, vol. 12, no. 03, 2015.

[17] Connor Schenck and Dieter Fox, "Visual closed-loop control for pouring liquids," in *ICRA*, 2017.

[18] Pieter Abbeel and Andrew Y Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*, 2004.

[19] Chelsea Finn, Sergey Levine, and Pieter Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *ICML*, 2016.

[20] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, 2008, vol. 8.

[21] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich, "Maximum margin planning," in *ICML*, 2006.

[22] Ashley D Edwards, *Perceptual Goal Specifications for Reinforcement Learning*, Ph.D. thesis, Georgia Institute of Technology, 2017.

[23] Umar Syed, Michael Bowling, and Robert E Schapire, "Apprenticeship learning using linear programming," in *ICML*, 2008.

[24] Daniel S. Brown, Yuchen Cui, and Scott Niekum, "Risk-aware active inverse reinforcement learning," in *CoRL*, 2018.

[25] Robert Cohn, Edmund Durfee, and Satinder Singh, "Comparing action-query strategies in semi-autonomous agents," in *AAAI*, 2011.

[26] Hsiao-Yu Tung, Adam W Harley, Liang-Kang Huang, and Katerina Fragkiadaki, "Reward learning from narrated demonstrations," in *CVPR*, 2018.

[27] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR*, vol. 37, no. 4-5, 2018.

[28] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine, "Variational inverse control with events: A general framework for data-driven reward definition," in *NIPS*, 2018.

[29] Ashley D Edwards, Srijan Sood, and Charles L Isbell Jr, "Cross-domain perceptual reward functions," *arXiv preprint arXiv:1705.09045*, 2017.

[30] Pierre Sermanet, Kelvin Xu, and Sergey Levine, "Unsupervised perceptual rewards for imitation learning," *arXiv preprint arXiv:1612.06699*, 2016.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[32] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[34] François Chollet et al., "Keras," https://keras.io, 2015.

[35] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *CASSP*, 2016.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[38] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[39] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and et al, "Language models are few-shot learners," in *NeurIPS*, 2020.

[40] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, 2016.

[41] Yaroslav Ganin and Victor Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*, 2015.

[42] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell, "Adversarial discriminative domain adaptation," in *CVPR*, 2017.