arXiv:1903.10869v1 [cs.CV] 23 Mar 2019

V2CNet: A Deep Learning Framework to Translate Videos to Commands for Robotic Manipulation

Journal Title
XX(X):1–15
©The Author(s) 2018
Reprints and permission:
TBD
DOI: 10.1177/ToBeAssigned
TBD

Anh Nguyen[1], Thanh-Toan Do[2], Ian Reid[2], Darwin G. Caldwell[1], Nikos G. Tsagarakis[1]

Abstract

We propose V2CNet, a new deep learning framework to automatically translate the demonstration videos to commands that can be directly used in robotic applications. Our V2CNet has two branches and aims at understanding the demonstration video in a fine-grained manner. The first branch has the encoder-decoder architecture to encode the visual features and sequentially generate the output words as a command, while the second branch uses a Temporal Convolutional Network (TCN) to learn the fine-grained actions. By jointly training both branches, the network is able to model the sequential information of the command, while effectively encodes the fine-grained actions. The experimental results on our new large-scale dataset show that V2CNet outperforms recent state-of-the-art methods by a substantial margin, while its output can be applied in real robotic applications. The source code and trained models will be made available.

Keywords

Video to Command, Fine-grained Video Captioning, Deep Learning, Learning from Demonstration

1. Introduction

While humans can effortlessly understand the actions and imitate the tasks by just *watching* someone else, making the robots to be able to perform actions based on observations of human activities is still a major challenge in robotics (Chrystopher L. Nehaniv, 2009). By understanding human actions, robots may acquire new skills, or perform different tasks, without the need for tedious programming. It is expected that the robots with these abilities will play an increasingly more important role in our society in areas such as assisting or replacing humans in disaster scenarios, taking care of the elderly, or helping people with everyday life tasks. Recently, this problem has been of great interest to researchers, many approaches have been proposed to tackle different tasks such as pouring water (Pastor et al., 2009), drawer opening (Rana et al., 2017), and multi-stage manipulation (Zhang et al., 2018).

In this work, we argue that there are two main capabilities that a robot must develop to be able to replicate human activities: *understanding* human actions, and *imitating* them. The imitation step has been widely investigated in robotics within the framework of learning from demonstration (LfD) (Argall et al., 2009). In particular, there are two main approaches in LfD that focus on improving the accuracy of the imitation process: kinesthetic teaching (Akgun et al., 2012) and motion capture (Koenemann et al., 2014). While the first approach needs the users to physically move the robot through the desired trajectories, the second approach uses a bodysuit or camera system to capture human motions. Although both approaches successfully allow a robot to imitate a human, the number of actions that the robot can learn is quite limited due to the need of using expensively physical systems (i.e., real robot, bodysuit, etc.) to capture the training data (Akgun et al., 2012; Koenemann et al., 2014).

The *understanding* step, on the other hand, receives more attention from the computer vision community. Two popular problems that receive a great deal of interest are video classification (Karpathy et al., 2014) and action recognition (Simonyan and Zisserman, 2014a). Recently, with the rise of deep learning, the video captioning task (Venugopalan et al., 2016) has become more feasible to tackle. Unlike the video classification or detection tasks which output only the class identity, the output of the video captioning task is a natural language sentence, which is potentially useful in robotic applications.

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy.
[2] Australian Centre for Robotic Vision, The University of Adelaide, SA 5005, Australia.

Corresponding author:
Anh Nguyen, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163, Genova, Italy.
Email: anh.nguyen@iit.it

In this paper, we propose a new deep learning framework that translates a demonstration video to a command which can directly be used in robotic applications. While the field of LfD focuses mainly on the imitation step, we focus on the understanding step, but our proposed method also allows the robot to perform useful tasks via the output commands. Our goal is to bridge the gap between computer vision and robotics, by developing a system that helps the robot understand human actions, and use this knowledge to complete useful tasks. Furthermore, since our method provides a meaningful way to let the robots understand human demonstrations by encoding the knowledge in the video, it can be integrated with any LfD techniques to improve the manipulation capabilities of the robot.

In particular, we cast the problem of translating videos to commands as a video captioning task: given a video, the goal is to translate this video to a command. Although we are inspired by the video captioning field, there are two key differences between our approach and the traditional video captioning task: (i) we use the *grammar-free* format in the captions for the convenience in robotic applications, and (ii) we aim at learning the commands through the demonstration videos that contain the *fine-grained* human actions. The use of fine-grained classes forms a more challenging problem than the traditional video captioning task since the fine-grained actions usually happen within a short duration, and there is usually ambiguous information between these actions. To effectively learn the fine-grained actions, unlike the traditional video and image captioning methods (Yao et al., 2015; You et al., 2016) that mainly investigate the use of *visual attention* to improve the result, we propose to use the Temporal Convolutional Networks (TCN) to explicitly classify the human actions. This TCN network can be considered as an *action attention* mechanism since it does not focus on the visual part of the data, but deeply encodes the fine-grained actions that humans are performing in the videos.

Based on the aforementioned observation, we design a network with two branches: a translation branch that interprets videos to commands using a RNN network, and an action classification branch that classifies the fine-grained human actions with a TCN network. The intuition of our design is that since both the classification and translation branches are jointly trained using a single loss function, the parameters of both branches are updated using the same gradient signal through backpropagation. Therefore, the classification branch will encourage the translation branch to generate the correct fine-grained action, which is the key information to understand human demonstrations. We experimentally demonstrate that by jointly train both branches, the translation results are significantly improved. Fig. 1 illustrates an overview of our video-to-command (V2CNet) deep network.

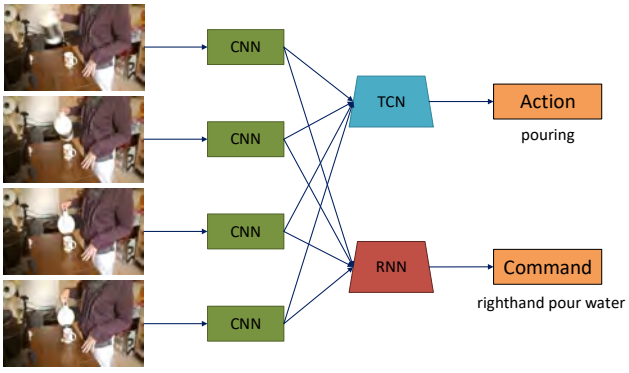


Fig. 1. An overview of our V2CNet architecture. The network is composed of two branches: a TCN branch to encode the actions, and a RNN branch to translate videos to commands.

To summarize, our main contributions are as follows:

- We reformulate the understanding step of the robot imitation problem as a fine-grained video captioning task, which potentially allows the robot to directly do the tasks from demonstration videos.
- We present a new fairly simple, yet effective deep architecture that translates videos to commands.
- We introduce a new large-scale dataset for the video-to-command task, which is suitable for deep learning approach.
- The methodology of our approach is validated in the real robotic systems. The source code and trained models will also be released.

Next, we review the related work in Section 2., then the detailed network architecture is described in Section 3.. In Section 4., we present the extensive experiments on our new IIT-V2C dataset and on the real robotic platforms. We discuss the advantages and limitations of our approach in Section 5., then conclude the paper in Section 6..

2. Related Work

Learning from Demonstration LfD techniques are widely used in the robotics community to teach the robots new skills based on human demonstrations (Argall et al., 2009). Two popular LfD approaches are kinesthetic teaching (Pastor et al., 2011; Akgun et al., 2012) and sensor-based demonstration (Calinon et al., 2009; Kruger et al., 2010). Recently, Koenemann et al. (2014) introduced a method to transfer complex whole-body human motions to a humanoid robot in real-time. Welschehold et al. (2016) proposed to transform human demonstrations into hand-object trajectories in order to adapt to robotic manipulation tasks. The advantage of LfD approaches is their abilities to let the robots accurately repeat human motions, however, it is difficult to expand LfD techniques to a large number

of tasks since the training process is usually designed for a specific task or needs training data from the real robotic systems (Akgun et al., 2012).

Action Representation From a computer vision point of view, Aksoy et al. (2016) represented the continuous human actions as "semantic event chains" and solved the problem as an activity detection task. Yang et al. (2015) proposed to learn manipulation actions from unconstrained videos using CNN and grammar based parser. However, they need an explicit representation of both the objects and grasping types to generate command sentences. Lee et al. (2013) captured the probabilistic activity grammars from the data for imitation learning. Ramirez-Amaro et al. (2015) extracted semantic representations from human activities in order to transfer skills to the robot. Recently, Plappert et al. (2017) introduced a method to learn bidirectional mapping between human motion and natural language with RNN. In this paper, we propose to directly learn the commands from the demonstration videos without any prior knowledge. Our method takes advantage of CNN to extract robust features, and RNN to model the sequences, while being easily adapted to any human activity.

Command Understanding Currently, commands are widely used to communicate and control real robotic systems. However, they are usually carefully programmed for each task. This limitation means programming is tedious if there are many tasks. To automatically allow the robot to understand the commands, Tellex et al. (2011) formed this problem as a probabilistic graphical model based on the semantic structure of the input command. Similarly, Guadarrama et al. (2013) introduced a semantic parser that used both natural commands and visual concepts to let the robot execute the task. While we retain the concepts of (Tellex et al., 2011; Guadarrama et al., 2013), the main difference in our approach is that we directly use the grammar-free commands from the translation module. This allows us to use a simple similarity measure to map each word in the generated command to the real command that can be used on the real robot.

Video Captioning With the rise of deep learning, the video captioning problem becomes more feasible to tackle. Donahue et al. (2014) introduced a first deep learning framework that can interpret an input video to a sentence. The visual features were first extracted from the video frames then fed into a LSTM network to generate the video captions. Recently, Yu et al. (2015) introduced a new hierarchical RNN to generate one or multiple sentences to describe a video. Venugopalan et al. (2016) proposed a sequence-to-sequence model to generate the video captions from both RGB and optical flow images. The authors in (Yao et al., 2015; Ramanishka et al., 2017) investigated the use of visual attention mechanism for this problem. Similarly, Pan et al. (2017) proposed to learn semantic attributes from the image then incorporated this information into a LSTM network. In this work, we cast the problem of translating videos to commands as a video captioning task to build on the strong state the art in computer vision. However, unlike (Yao et al., 2015; Ramanishka et al., 2017) that explore the visual attention mechanism to improve the result, we focus on the fine-grained actions in the video. Our hypothesis is based on the fact that the fine-grained action is the key information in the video-to-command task, hence plays a significant contribution to the results.

Reinforcement/Meta Learning Recently, reinforcement learning and meta-learning techniques are also widely used to solve the problem of learning from human demonstrations. Rhinehart and Kitani (2017) proposed a method to predict the outcome of the human demonstrations from a scene using inverse reinforcement learning. Stadie et al. (2017) learned a reward function based on human demonstrations of a given task in order to allow the robots to execute some trials, then maximize the reward to reach the desired outcome. With a different viewpoint, Finn et al. (2017) proposed an one-shot visual imitation learning method to let the robot perform the tasks from just one single demonstration. More recently, Yu et al. (2018) presented a new approach for one-shot learning by fusing human and robot demonstration data to build up prior knowledge through meta-learning. The main advantage of meta-learning approach is it allows the robot to replicate human actions from just one or few demonstrations, however handling domain shift and the dependence on the data from the real robotic system are still the major challenges in this approach.

3. Translating Videos to Commands

We start by formulating the problem and briefly describing three basic components (recurrent neural networks, visual feature extraction, and word embedding) that are used in our design. We then present in details the architecture of our video-to-command network (V2CNet) that simultaneously translates the input videos to robotic commands and classifies the fine-grained actions.

3.1. Problem Formulation

In order to effectively generate both the output command and understand the fine-grained action in the video, we cast the problem of translating videos to commands as a video captioning task, and use a TCN network to classify the actions. Both tasks use the same input feature and are jointly trained using a single multi-task loss function. In particular, the input video is considered as a list of frames, presented by a sequence of features $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ from each frame. The output command is presented as a sequence of word vectors $\mathbf{Y}^w = (\mathbf{y}_1^w, \mathbf{y}_2^w, ..., \mathbf{y}_m^w)$, in which each vector $\mathbf{y}^w$ represents one word in the dictionary D. Each input

video also has an action groundtruth label, represented as an one-hot vector $\mathbf{Y}^a \in \mathbb{R}^{|C|}$, where C is the number of action classes, to indicate the index of the truth action class. Our goal is to simultaneously find for each sequence feature $\mathbf{X}_i$ its most probable command $\mathbf{Y}_i^w$, and its best suitable action class $\mathbf{Y}_i^a$. In practice, the number of video frames n is usually higher than the number of words m. To make the problem more suitable for robotic applications, we use a dataset that contains mainly human demonstrations and assume that the command is in grammar-free format.

3.2. Background

3.2.1. Recurrent Neural Networks

LSTM LSTM is a popular RNN since it can effectively model the long-term dependencies from the input data through its gating mechanism. The LSTM network takes an input $\mathbf{x}_t$ at each time step t, and computes the hidden state $\mathbf{h}_t$ and the memory cell state $\mathbf{c}_t$ as follows:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{g}_t &= \phi(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}h_{t-1} + \mathbf{b}_g) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \phi(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where $\odot$ represents element-wise multiplication, the function $\sigma : \mathbb{R} \mapsto [0, 1], \sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmod non-linearity, and $\phi : \mathbb{R} \mapsto [-1, 1], \phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic tangent non-linearity. The parameters $\mathbf{W}$ and $\mathbf{b}$ are trainable weight and bias of the LSTM network. With this gating mechanism, the LSTM network can remember or forget information for long periods of time, while is still robust against the vanishing gradient problem. In practice, the LSTM network is straightforward to train end-to-end and is widely used in many problems (Donahue et al., 2014; Nguyen et al., 2017a; Ramanishka et al., 2017; Nguyen et al., 2018a).

GRU Another popular RNN is Gated Recurrent Unit (GRU) (Cho et al., 2014). The main advantage of the GRU network is that it uses less computational resources in comparison with the LSTM network, while achieves competitive performance. Unlike the standard LSTM cell, in a GRU cell, the update gate controls both the input and forget gates, while the reset gate is applied before the nonlinear transformation as follows:

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r) \\
\mathbf{z}_t &= \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z) \\
\tilde{\mathbf{h}}_t &= \phi(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h \\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t
\end{aligned} \tag{2}$$

where $\mathbf{r}_t$, $\mathbf{z}_t$, and $\mathbf{h}_t$ represent the reset, update, and hidden gate of the GRU cell, respectively.

3.2.2. Command Embedding

Since a command is a list of words, we have to represent each word as a vector for computation. There are two popular techniques for word representation: *one-hot* encoding and *word2vec* (Mikolov et al., 2013) embedding. Although the one-hot vector is high dimensional and sparse since its dimensionality grows linearly with the number of words in the vocabulary, it is straightforward to use this embedding in the video captioning task. In this work, we choose the one-hot encoding technique as our word representation since the number of words in our dictionary is relatively small. The one-hot vector $\mathbf{y}^w \in \mathbb{R}^{|D|}$ is a binary vector with only one non-zero entry indicating the index of the current word in the vocabulary. Formally, each value in the one-hot vector $\mathbf{y}^w$ is defined by:

$$\mathbf{y}^w(i) = \begin{cases} 1, & \text{if } i = ind(\mathbf{y}^w) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $ind(\mathbf{y}^w)$ is the index of the current word in the dictionary D. In practice, we add an extra word EOC to the dictionary to indicate the end of command sentences.

3.2.3. Visual Features

As the standard practice in the video captioning task (Venugopalan et al., 2016; Ramanishka et al., 2017), the visual feature from each video frame is first extracted offline then fed into the captioning module. This step is necessary since we usually have a lot of frames from the input videos. In practice, we first sample n frames from each input video in order to extract deep features from the images. The frames are selected uniformly with the same interval if the video is too long. In case the video is too short and there are not enough n frames, we create an artificial frame from the mean pixel values of the ImageNet dataset (Russakovsky et al., 2015) and pad this frame at the end of the list until it reaches n frames. We then use the state-of-the-art CNN to extract deep features from these input frames. Since the visual features provide the key information for the learning process, three popular CNN are used in our experiments: VGG16 (Simonyan and Zisserman, 2014b), Inception_v3 (Szegedy et al., 2016), and ResNet50 (He et al., 2016).

Specifically, for the VGG16 network, the features are extracted from its last fully connected `fc2` layer. For the Inception_v3 network, we extract the features from its `pool_3:0` tensor. Finally, we use the features from `pool5` layer of the ResNet50 network. The dimension of the extracted features is 4096, 2048, 2048, for the VGG16, Inception_v3, and ResNet50 network, respectively. All these CNN are pretrained on ImageNet dataset for image classifications. We notice that the names of the layers

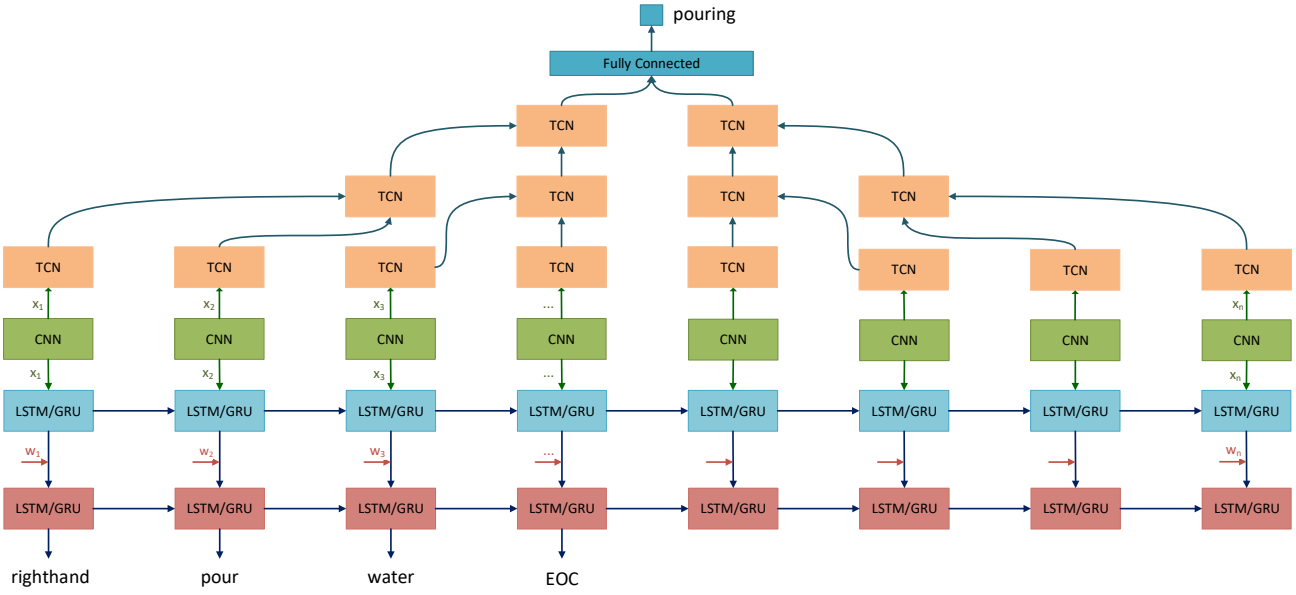


Fig. 2. A detailed illustration of our V2CNet architecture. The network is composed of two branches: a classification branch and a translation branch. The input for both branches is the visual features extracted by CNN. The classification branch uses the input features to learn the action through a TCN network. The translation branch has the encoder-decoder architecture with two LSTM/GRU layers. The first LSTM/GRU layer is used to encode the visual features, then the input words are combined with the output of the first LSTM/GRU layer and fed into the second LSTM/GRU layer in order to sequentially generate the output words as a command.

we mention here are based on the Tensorflow (Abadi et al., 2015) implementation of these networks.

3.3. Architecture

3.3.1. Overview To simultaneously generate the command sentence and classify the fine-grained actions, we design a deep network with two branches: a classification branch and a translation branch. The classification branch is a TCN network that handles the human action, while the translation branch is a RNN with encoder-decode architecture to encode the visual features and sequentially generate the output words as a command. Both branches share the input visual features from the video frames and are trained end-to-end together using a single multi-task loss function. Our intuition is that since the translation branch has to generate the command from a huge space of words, it may not effectively encode the fine-grain actions. Therefore, we integrate the classification branch which is trained directly on the smaller space of action classes to ease the learning process. In practice, the parameters of both branches are updated using the same gradient signal, hence the classification branch will encourage the translation branch to generate the correct fine-grained action. Fig. 2 illustrates the details of our V2CNet network.

3.3.2. Fine-grained Action Classification We employ a TCN network in the classification branch to encode the temporal information of the video in order to classify the fine-grained actions. Unlike the popular 2D convolution that operates on the 2D feature map of the image, the TCN network performs the convolution across the time axis of the data. Since each video is represented as a sequence of frames and the fine-grained action is the key information for the learning process, using the TCN network to encode this information is a natural choice. Recent work showed that the TCN network can further improve the results in many tasks such as action segmentation and detection (Sun et al., 2015; Lea et al., 2017).

The input of the classification branch is a set of visual features extracted from the video frames by a pretrained CNN. The output of this branch is a classification score that indicates the most probable action for the input video. In practice, we build a TCN with three convolutional layers to classify the fine-grained actions. After each convolutional layer, a non-linear activation layer (ReLU) is used, followed by a max pooling layer to gradually encode the temporal information of the input video. Here, we notice that instead of performing the max pooling operation in the 2D feature map, the max pooling of TCN network is performed across the time axis of the video. Finally, the output of the third convolutional layer is fed into a fully connected layer with 256 neurons, then the last layer with only 1 neuron is used to regress the classification score for the current video.

More formally, given the input feature vector $\mathbf{X}$, three convolutional layers $\mathbf{\Phi}_c$ of the TCN network are defined as

follows to encode the temporal information across the time axis of the input demonstration video:

$$\begin{aligned}\mathbf{\Phi}_{c0}(\mathbf{W}_{c0},\mathbf{b}_{c0}) &= \mathbf{W}_{c0}\mathbf{X}+\mathbf{b}_{c0}\\ \mathbf{\Phi}_{c1}(\mathbf{W}_{c1},\mathbf{b}_{c1}) &= \mathbf{W}_{c1}(\text{MaxPool}(\text{ReLU}(\mathbf{\Phi}_{c0}))+\mathbf{b}_{c1}\\ \mathbf{\Phi}_{c2}(\mathbf{W}_{c2},\mathbf{b}_{c2}) &= \mathbf{W}_{c2}(\text{MaxPool}(\text{ReLU}(\mathbf{\Phi}_{c1}))+\mathbf{b}_{c2}\end{aligned} \tag{4}$$

then the third convolutional layer $\mathbf{\Phi}_{c2}$ is fed into a fully connected layer $\mathbf{\Phi}_{f0}$ as follows:

$$\begin{aligned}\mathbf{\Phi}_{f0}(\mathbf{W}_{f0},\mathbf{b}_{f0}) &= \mathbf{W}_{f0}(\text{ReLU}(\mathbf{\Phi}_{c2}))+\mathbf{b}_{f0}\\ \mathbf{\Phi}_{f1}(\mathbf{W}_{f1},\mathbf{b}_{f1}) &= \mathbf{W}_{f1}\mathbf{\Phi}_{f0}+\mathbf{b}_{f1}\end{aligned} \tag{5}$$

where $\mathbf{W}_c$, $\mathbf{b}_c$ and $\mathbf{W}_f$, $\mathbf{b}_f$ are the weight and bias of the convolutional and fully connected layers. In practice, the filter parameters of three convolutional layers $\mathbf{\Phi}_{c0}$, $\mathbf{\Phi}_{c1}$, $\mathbf{\Phi}_{c2}$ are empirically set to 2048, 1024, and 512, respectively. Note that, the ReLU activation is used in the first fully-connected $\mathbf{\Phi}_{f0}$ layer, while there is no activation in the last fully connected layer $\mathbf{\Phi}_{f1}$ since we want this layer outputs a probability of the classification score for each fine-grained action class.

3.3.3. Command Generation In parallel with the classification branch, we build a translation branch to generate the command sentence from the input video. The architecture of our translation branch is based on the encoder-decoder scheme (Venugopalan et al., 2016; Ramanishka et al., 2017), which is adapted from the popular sequence to sequence model (Sutskever et al., 2014) in machine translation. Although recent approaches use the visual attention mechanism (Yao et al., 2015; Ramanishka et al., 2017) or hierarchical RNN (Yu et al., 2015) to improve the results, our translation branch solely relies on the neural architecture. However, since the translation branch is jointly trained with classification branch, it is encouraged to output the correct fine-grained action as learned by the classification branch. We experimentally show that by simultaneously training both branches, the translation accuracy is significantly improved over the state of the art.

In particular, the translation branch also uses the visual features extracted by a pretrained CNN as in the classification branch. These features are encoded in the first RNN layer to create the encoder hidden state. The input words are then combined with the hidden state of the first RNN layer, and fed into the second RNN layer. This layer decodes its input and sequentially generates a list of words as the output command. More formally, given an input sequence of features $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n)$, we want to estimate the conditional probability for an output command $\mathbf{Y}^w=(\mathbf{y}_1^w,\mathbf{y}_2^w,...,\mathbf{y}_m^w)$ as follows:

$$P(\mathbf{y}_1^w,...,\mathbf{y}_m^w|\mathbf{x}_1,...,\mathbf{x}_n)=\prod_{i=1}^{m}P(\mathbf{y}_i^w|\mathbf{y}_{i-1}^w,...,\mathbf{y}_1^w,\mathbf{X}) \tag{6}$$

Since we want a generative model that encodes a sequence of features and produces a sequence of words in order as a command, the LSTM/GRU is well suitable for this task. Another advantage of LSTM/GRU is that they can model the long-term dependencies in the input features and the output words. In practice, we use LSTM and GRU as our recurrent neural network, while the input visual features are extracted from the VGG16, Inception_v3, and ResNet50 network, respectively.

In the encoding stage, the first LSTM/GRU layer converts the visual features $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n)$ to a list of hidden state vectors $\mathbf{H}^e=(\mathbf{h}_1^e,\mathbf{h}_2^e,...,\mathbf{h}_n^e)$ (using Equation 1 for the LSTM network or Equation 2 for the GRU network). Unlike Venugopalan et al. (2014) which takes the average of all n hidden state vectors to create a fixed-length vector, we directly use each hidden vector $\mathbf{h}_i^e$ as the input $\mathbf{x}_i^d$ for the second decoder layer. This allows the smooth transaction from the visual features to the output commands without worrying about the harsh average pooling operation, which can lead to the loss of temporal structure underlying the input video.

In the decoding stage, the second LSTM/GRU layer converts the list of hidden encoder vectors $\mathbf{H}^e$ into the sequence of hidden decoder vectors $\mathbf{H}^d$. The final list of predicted words $\hat{\mathbf{Y}}$ is achieved by applying a softmax layer on the output $\mathbf{H}^d$ of the LSTM/GRU decoder layer. In this way, the LSTM/GRU decoder layer sequentially generates a conditional probability distribution for each word of the output command given the encoded features representation and all the previously generated words. In practice, we preprocess the data so that the number of input words m is equal to the number of input frames n. For the input video, this is done by uniformly sampling n frames in the long video, or padding the extra frame if the video is too short. Since the number of words m in the input commands is always smaller than n, we pad a special empty word to the list until we have n words.

3.3.4. Multi-Task Loss We train the V2CNet using a joint loss function for both the classification and translation branches as follows:

$$L=L_{cls}+L_{trans} \tag{7}$$

where the L_{cls} loss is defined in the classification branch for fine-grained action classification, and L_{trans} is defined in the RNN branch for command generation.

Specially, L_{cls} is the sigmoid cross entropy loss over the groundtruth action classes C, and is defined as follows:

$$L_{cls} = -\sum_{i=1}^{C} \mathbf{y}_i^a \log(\hat{\mathbf{y}}_i^a) \tag{8}$$

where $\mathbf{y}^a$ is the groundtruth action label of the current input video, and $\hat{\mathbf{y}}^a$ is the predicted action output of the classification branch of the network.

We follow the process in (Donahue et al., 2014) to compute the L_{trans} loss for the translation branch. Let $\mathbf{z}_t$ denote the output of each cell at each time step t in the second LSTM/GRU layer in the translation branch. This output is then passed through a linear prediction layer $\hat{\mathbf{y}}_t = \mathbf{W}_z \mathbf{z}_t + \mathbf{b}_z$, and the predicted distribution $P(\mathbf{y}_t)$ is computed by taking the softmax of $\hat{\mathbf{y}}_t$ as follows:

$$P(\mathbf{y}_t = \mathbf{w}|\mathbf{z}_t) = \frac{\exp(\hat{\mathbf{y}}_{t,w})}{\sum_{w' \in D} \exp(\hat{\mathbf{y}}_{t,w'})} \tag{9}$$

where $\mathbf{W}_z$ and $\mathbf{b}_z$ are learned parameters, $\mathbf{w}$ is a word in the dictionary D. The L_{trans} loss is then computed as follows:

$$L_{trans} = \frac{1}{m} \sum_{t=1}^{m} P(\mathbf{y}_t = \mathbf{w}|\mathbf{z}_t) \tag{10}$$

Intuitively, Equation 10 computes the softmax loss at each word of the current output command given all the previous generated words.

3.3.5. Training and Inference

During the training phase, the classification branch uses the visual features to learn the fine-grained actions using the TCN network. In parallel with the classification branch, the translation branch also receives the input via its first LSTM/GRU layer. In particular, at each time step t, the input feature $\mathbf{x}_t$ is fed to an LSTM/GRU cell in the first LSTM/GRU layer along with the previous hidden state $\mathbf{h}_{t-1}^e$ to produce the current hidden state $\mathbf{h}_t^e$. After all the input features are exhausted, the word embedding and the hidden states of the first LSTM/GRU layer are fed to the second LSTM/GRU layer. This layer converts the inputs into a sequence of words by maximizing the log-likelihood of the predicted word. This decoding process is performed sequentially for each word until the network generates the EOC token. Since both the classification and translation branches are jointly trained using a single loss function, the weight and bias parameters of both branches are updated using the same gradient signal through backpropagation.

During the inference phase, the input for the network is only the visual features of the testing video. The classification branch uses these visual features to generate the probabilities for all action classes. The final action class is chosen from the class with the highest classification score. Similarly, the visual features are fed into two LSTM/GRU layers to sequentially generate the output words as the command. Note that, unlike the training process, during inference the input for the second LSTM/GRU layer in the translation branch is only the hidden state of the first LSTM/GRU layer. The final command sentence is composed of the first generated word and the last word before the EOC token.

4. Experiments

4.1. Dataset

Recently, many datasets have been proposed in the video captioning field (Xu et al., 2016). However, these datasets only provide general descriptions of the video and there is no detailed understanding of the action. The groundtruth captions are also written using natural language sentences which can not be used directly in robotic applications. Motivated by these limitations, we introduce a new *video-to-command* (IIT-V2C) dataset which focuses on *fine-grained* action understanding (Lea et al., 2016). Our goal is to create a new large-scale dataset that provides fine-grained understanding of human actions in a grammar-free format. This is more suitable for robotic applications and can be used with deep learning methods.

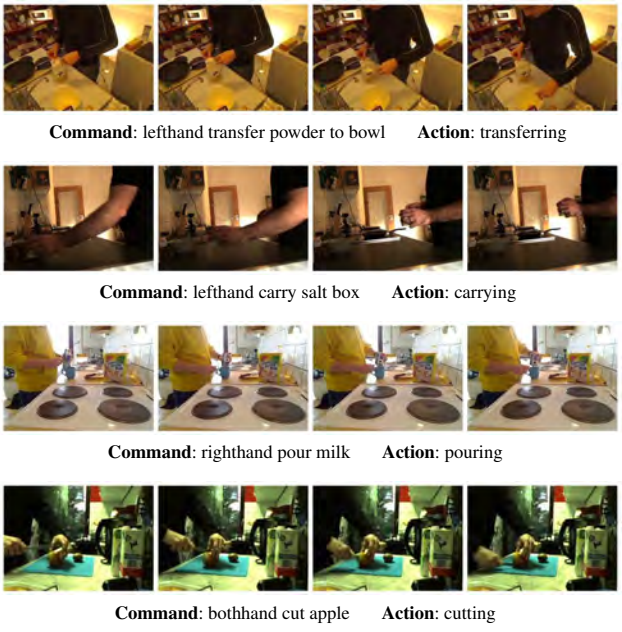


Fig. 3. Example of human demonstration clips and labeled groundtruths for the command sentence and fine-grained action class in our IIT-V2C dataset. The clips were recorded from challenging viewpoints with different lighting conditions.

Video annotation Since our main purpose in this work is to develop a framework that can be used by real robots

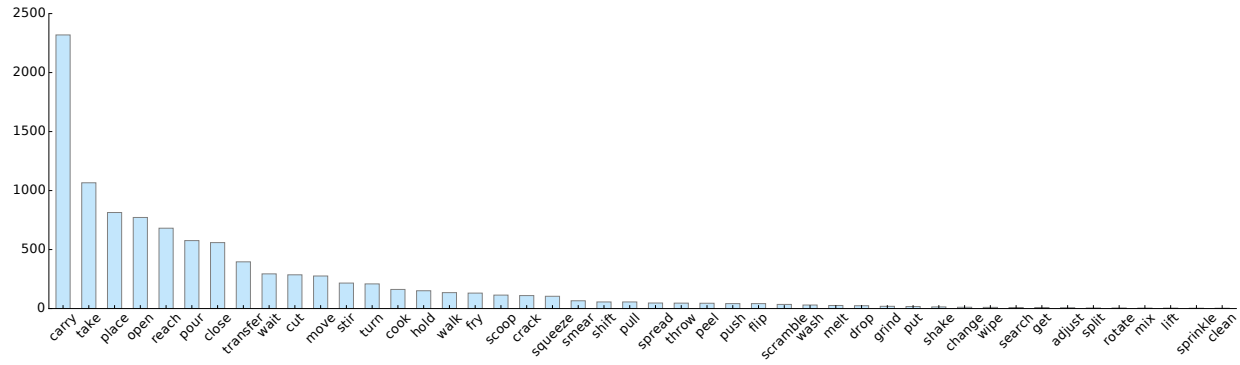


Fig. 4. The distribution of the fine-grained action classes in our IIT-AFF dataset.

for manipulation tasks, we are only interested in the videos that have human demonstrations. To this end, the raw videos in the Breakfast dataset (Kuehne et al., 2014) are best suited to our purpose since they were originally designed for activity recognition. We only reuse the raw videos from the Breakfast dataset and manually segment each video into short clips in a fine granularity level. Each short clip is then annotated with a *command sentence* in grammar-free that describes the current human action. For each command sentence, we use the Stanford POS Tagger (Toutanova et al., 2003) to automatically extract the verb from the command. This extracted verb is then considered as the fine-grained *action groundtruth* for the current video.

Dataset statistics Overall, we reuse 419 videos from the Breakfast dataset. These videos were captured when humans performed cooking tasks in different kitchens, then encoded with 15 frames per second. The resolution of demonstration videos is 320×240. We segment each video (approximately $2-3$ minutes long) into around $10-50$ short clips (approximately $1-15$ seconds long), resulting in $11,000$ unique short videos. Each short video has a single command sentence that describes human actions. From the groundtruth command sentence, we extract the verb as the action class for each video, resulting in an action set with 46 classes (e.g., cutting, pouring, etc.). Fig. 3 shows some example frames of groudtruth command sentences and action classes in our new dataset. In Fig. 4, the distribution of the fine-grained action classes is also presented. Although our new-form dataset is characterized by its grammar-free property for the convenience in robotic applications, it can easily be adapted to classical video captioning task by adding the full natural sentences as the new groundtruth for each video.

4.2. Evaluation, Baseline, and Implementation

Evaluation Metric We use the standard evaluation metrics in the video captioning field (Xu et al., 2016) (Bleu, METEOR, ROUGE-L, and CIDEr) to report the translation results of our V2CNet. By using the same evaluation metrics, we can directly compare our results with the recent state of the art in the video captioning field.

Baseline The translation results of our V2CNet are compared with the following state of the art in the field of video captioning: S2VT (Venugopalan et al., 2016), SGC (Ramanishka et al., 2017), and SCN (Gan et al., 2017). In S2VT, the authors used the encoder-decoder architecture with LSTM to encode the visual features from RGB images (extracted by ResNet50 or VGG16) and optical flow images (extracted by AlexNet (Krizhevsky et al., 2012)). In SGC, the authors also used the encoder-decoder architecture and LSTM, however, this work integrated a saliency guided method as the visual attention mechanism, while the visual features are from the Inception network. The authors in SCN (Gan et al., 2017) first combined the visual features from ResNet and temporal features from C3D (Tran et al., 2015) network, then extracted semantic concepts (i.e., tags) from the video frames. All these features were learned in a LSTM as a semantic recurrent neural network. Finally, we also compare the V2CNet results with our early work (EDNet (Nguyen et al., 2018b)). The key difference between EDNet and V2CNet is the EDNet does not use the TCN network to jointly train the fine-grained action classification branch and the translation branch as in the V2CNet. For all methods, we use the code provided by the authors of the associated papers for the fair comparison.

Implementation We use 512 hidden units in both LSTM and GRU in our implementation. The first hidden state of LSTM/GRU is initialized uniformly in $[-0.1, 0.1]$. We set the number of frames for each input video at $n = 30$. Subsequently, we consider each command has maximum 30 words. If there are not enough 30 frames/words in the input video/command, we pad the mean frame/empty word at the end of the list until it reaches 30. The mean frame is composed of pixels with the same mean RGB value from the ImageNet dataset (i.e., $(104, 117, 124)$). We use 70% of the IIT-V2C dataset for training and the remaining 30%

Table 1. The translation results on IIT-V2C dataset.

	RNN	Feature	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	CIDEr
S2VT (Venugopalan et al., 2016)	LSTM	VGG16, AlexNet	0.383	0.265	0.201	0.159	0.183	0.382	1.431
S2VT (Venugopalan et al., 2016)	LSTM	ResNet50, AlexNet	0.397	0.280	0.219	0.177	0.196	0.401	1.560
SGC (Ramanishka et al., 2017)	LSTM	Inception	0.370	0.256	0.198	0.161	0.179	0.371	1.422
SCN (Gan et al., 2017)	LSTM	ResNet50, C3D	0.398	0.281	0.219	0.190	0.195	0.399	1.561
EDNet (Nguyen et al., 2018b)	LSTM	VGG16	0.372	0.255	0.193	0.159	0.180	0.375	1.395
		Inception	0.400	0.286	0.221	0.178	0.194	0.402	1.594
		ResNet50	0.398	0.279	0.215	0.174	0.193	0.398	1.550
	GRU	VGG16	0.350	0.233	0.173	0.137	0.168	0.351	1.255
		Inception	0.391	0.281	0.222	0.188	0.190	0.398	1.588
		ResNet50	0.398	0.284	0.220	0.183	0.193	0.399	1.567
V2CNet (ours)	LSTM	VGG16	0.391	0.275	0.212	0.174	0.189	0.393	1.528
		Inception	0.401	0.289	0.227	0.190	0.196	0.403	1.643
		ResNet50	**0.406**	**0.293**	**0.233**	**0.199**	**0.198**	**0.408**	**1.656**
	GRU	VGG16	0.389	0.267	0.208	0.172	0.186	0.387	1.462
		Inception	0.402	0.285	0.224	0.189	0.196	0.405	1.618
		ResNet50	0.403	0.288	0.226	0.191	0.196	0.403	1.596

for testing. During the training phase, we only accumulate the softmax losses of the real words to the total loss, while the losses from the empty words are ignored. We train all the variations of V2CNet for 300 epochs using Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0001, and the batch size is empirically set to 16. The training time for each variation of the V2CNet is around 8 hours on an NVIDIA Titan X GPU.

4.3. Translation Results

Table 1 summarizes the translation results on the IIT-V2C dataset. This table clearly shows that our V2CNet with ResNet50 feature achieves the highest performance in all metrics: Blue_1, Blue_2, Blue_3, Blue_4, METEOR, ROUGE_L, and CIDEr. Our proposed V2CNet also outperforms recent the state-of-the-art methods in the video captioning field (S2VT, SGC, and SCN) by a substantial margin. In particular, the best CIDEr score of V2CNet is 1.656, while the CIDEr score of the closest runner-up SCN is only 1.561. Furthermore, compared with the results by EDNet, V2CNet also shows a significant improvement in all experiments when different RNN types (i.e., LSTM or GRU) or visual features (i.e., VGG16, Inception, ResNet50) are used. These results demonstrate that by jointly learning both the fine-grained actions and the commands, the V2CNet can effectively encode both the visual features to generate the commands while is able to understand the fine-grained action from the demonstration videos. Therefore, the translation results are significantly improved.

Overall, we have observed a consistent improvement of our proposed V2CNet over EDNet in all variation setups. The improvement is most significant when the VGG16 feature is used. In particular, while the results of EDNet using VGG16 feature with both LSTM and GRU networks are relatively low, our V2CNet shows a clear improvement in these cases. For example, the Blue_1 scores of EDNet using VGG16 feature are only 0.372 and 0.350 with the LSTM and GRU network respectively, while with the same setup, the V2CNet results are 0.391 and 0.389. We also notice that since EDNet only focuses on the translation process, it shows a substantial gap in the results between the VGG16 feature and two other features. However, this gap is gradually reduced in the V2CNet results. This demonstrates that the fine-grained action information in the video plays an important role in the task of translating videos to commands, and the overall performance can be further improved by learning the fine-grained actions.

From this experiment, we notice that there are three main factors that affect the results: the translation architecture, the input visual feature, and the external mechanism such as visual or action attention. While the encoder-decoder architecture is widely used to interpret videos to sentences, recent works focus on exploring the use of robust features and attention mechanism to improve the results. Since the IIT-V2C dataset contains mainly the fine-grained human demonstrations, the visual attention mechanism (such as in SGC architecture) does not perform well as in the normal video captioning task. On the other hand, the action

attention mechanism and the input visual features strongly affect the final results. Our experiments show that the use of TCN network as the action attention mechanism clearly improves the translation results. In general, we note that the temporal information of the video plays an important role in this tasks. By extracting this information offline (e.g., from optical flow images as in S2VT, or with C3D network as in SCN), or learning it online as in our V2CNet, the translation results can be further improved.

Fig. 5 shows some comparisons between the commands generated by our V2CNet and other methods, compared with the groundtruth on the test videos of the IIT-V2C dataset. In general, our V2CNet is able to generate good predictions in many examples, while the results of other methods are more variable. In comparison with EDNet, V2CNet shows more generated commands with the correct fine-grained actions. We notice that in addition to the generated commands that are identical with the groundtruth, many other output commands are relevant. These qualitative results show that our V2CNet can further improve the translation results. However, there are still many wrong predictions in the results of all the methods. Since the IIT-V2C dataset contains the fine-grained actions, while the visual information is also difficult (e.g, the relevant objects are small and usually are covered by the hand, etc.). This makes the problem of translating videos to commands is more challenging than the normal video captioning task since the network has to rely on the minimal information to predict the command sentence.

To conclude, the extensive experiments using different feature extraction methods and RNNs show that our V2CNet successfully encodes the visual features and generates the associated command for each video. Our proposed method also outperforms recent state of the art by a substantial margin. The key idea that improves the translation results is the integration of the TCN network into the traditional encoder-decoder architecture to effectively learn the fine-grained actions in the video.

4.4. Ablation Studies

Although the traditional captioning metrics such as Bleu, METEOR, ROUGE_L, and CIDEr give us the quantitative evaluation about the translation results, they use all the words in the generated commands for the evaluation, hence do not provide details about the accuracy of the fine-grained human actions. To analyze the prediction accuracy of the fine-grained actions, we conduct the following experiments: For both EDNet and V2CNet, we use the LSTM network and the visual features from the VGG16, Inception, and ResNet to generate both the commands from the translation branch, and the action class from the classification branch. The predicted actions of the translation branch are then automatically extracted by using the Stanford POS Tagger (Toutanova et al., 2003) to select the verb from the generated commands, while the classification branch gives us directly the fine-grained action class. For each variation of the networks, we report the success rate of the predicted output as the percentage of the correct predictions over all the testing clips. Intuitively, this experiment evaluates the accuracy of the fine-grained action prediction when we consider the translation branch also outputs the fine-grained action classes.

Table 2. The fine-grained action classification success rate.

	Feature	Success Rate
EDNet (Nguyen et al., 2018b)	VGG16	30.17%
	Inception	32.88%
	ResNet50	32.71%
V2CNet (translation branch)	VGG16	31.75%
	Inception	34.41%
	ResNet50	**34.81**%
V2CNet (classification branch)	VGG16	31.94%
	Inception	34.52%
	ResNet50	34.69%

Table 2 summaries the success rate of the fine-grained action classification results. Overall, we observe a consistent improvement of V2CNet over EDNet in all experiments using different visual features. In particular, when the actions are extracted from the generated commands of the translation branch in V2CNet, we achieve the highest success rate of 34.81% with the ResNet50 features. This is a 2.1% improvement over EDNet with ResNet50 features, and 4.64% over EDNet with VGG16 features. While Table 2 shows that the V2CNet clearly outperforms EDNet in all variation setups, the results of the translation branch and classification branch of V2CNet are a tie. These results demonstrate that the use of the TCN network in the classification branch is necessary for improving both the translation and classification accuracy. However, the overall classification success rate is relatively low and the problem of translating videos to commands still remains very challenging since it requires the understanding of the fine-grained actions.

4.5. Robotic Applications

Similar to (Yang et al., 2015), our long-term goal is to develop a framework that allows the robot to perform various manipulation tasks by just "*watching*" the input video. While the V2CNet is able to interpret a demonstration video to a command, the robot still needs more information such as scene understanding (e.g., object affordance, grasping frame, etc.), and trajectory

Fig. 5. Example of translation results of our V2CNet and other methods in comparison with the groundtruth (**GT**).

planner to complete a manipulation task. In this work, we build a framework based on three basic components: action understanding, affordance detection, and trajectory generation. In practice, the proposed V2CNet is first used to let the robot understand the human demonstration from a video, then the AffordanceNet (Do et al., 2018) which is trained on IIT-AFF dataset (Nguyen et al., 2017b) is used to localize the object affordances and the grasping frames. Finally, the motion planner is used to generate the trajectory for the robot in order to complete the manipulation tasks.

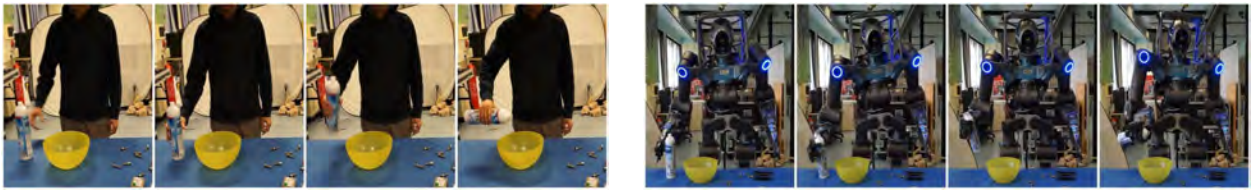
We conduct the experiments using two robotic platforms: WALK-MAN (Tsagarakis et al., 2017) humanoid robot and UR5 arm. WALK-MAN is a full-size humanoid robot with 31 DoF and two underactuated hands. Each hand has five fingers but only 1 DoF, and is controlled by one single motor. Therefore, it can only simultaneously generate the open-close grasping motion of all five fingers. We use XBotCore software architecture (Muratore et al., 2017) to handle the communication with the robot, while the full-body motion is planned by OpenSoT library (Rocchi et al., 2015). The vision system of WALK-MAN is equipped with a Multisense SL camera, while in UR5 arm we use a RealSense camera to provide visual data for the robot. Due to the hardware limitation of the end-effector in both robots, we only demonstrate the tasks that do not require precise skills of the fingers (e.g., cutting, hammering), and assume that the motion planning library can provide feasible solutions during the execution.

For each task presented by a demonstration video, V2CNet first generates a command sentence, then based on this command the robot uses its vision system to find relevant objects and plan the actions. Fig. 6 shows some tasks performed by WALK-MAN and UR5 arm using our framework. For a simple task such as "righthand grasp bottle", the robots can effectively repeat the human action through the command. Since the output of our translation module is in grammar-free format, we can directly map each word in the command sentence to the real robot command. In this way, we avoid using other modules as in (Tellex et al., 2011) to parse the natural command into the one that uses in the real robot. The visual system also plays an important role in our framework since it provides the target frames for the planner. Using our approach, the robots can also complete long manipulation tasks by stacking a list of demonstration videos in order for the translation module. Note that, for the long manipulation tasks, we assume that the ending state of one task will be the starting state of the next task. Overall, the robots successfully perform various tasks such as grasping, pick and place, or pouring. Our experimental video is available at: https://sites.google.com/site/v2cnetwork.

5. Discussion

In this work, we divide the robot imitation task into two steps: the understanding step and the imitation step. We form the understanding step as a fine-grained video

(a) Pick and place task using WALK-MAN



(b) Pouring task using WALK-MAN

(c) Pick and place task using UR5 arm

Fig. 6. Example of manipulation tasks performed by WALK-MAN and UR5 arm using our proposed framework: **(a)** Pick and place task using WALK-MAN, **(b)** Pouring task using WALK-MAN, and **(c)** Pick and place task using UR5 arm. The frames from human instruction videos are on the left side, while the robot performs actions on the right side. We notice that there are two sub-tasks (i.e., two commands) in these tasks: grasping the object and manipulating it. More illustrations can be found in the supplemental video.

captioning task and solve it as a visual translation problem. From the extensive experiments on the IIT-V2C dataset, we have observed that the translation accuracy not only depends on the visual information of each frame but also depends on the temporal information across the video. By using the TCN network to encode the temporal information, we achieved a significant improvement over the state of the art. Despite this improvement, we acknowledge that this task remains very challenging since it requires the fine-grained understanding of human actions, which is still an unsolved problem in computer vision (Lea et al., 2016).

From a robotic point of view, by explicitly representing the human demonstration as a command then combining it with the vision and planning module, we do not have to handle the domain shift problem (Yu et al., 2018) in the real experiment. Another advantage of this approach is we can reuse the strong state-of-the-art results from the vision and planning fields. However, its main drawback is the reliance on the accuracy of each component, which may become the bottleneck in real-world applications. For example, our framework currently relies solely on the vision system to provide the grasping frames and other useful information for the robot. Although recent advances in deep learning allow us to have powerful recognition systems, these methods are still limited by the information presented in the training data, and cannot provide fully semantic understanding of the scene (Do et al., 2018).

Due to the complexity of the whole robotic framework which requires many steps, our robotic experiments so far are qualitative. Since we design the framework as separated modules, the accuracy of the whole system can be improved by upgrading each module. From the robotic experiments, we notice that while the vision and planning modules can produce reasonable results, the translation module is still the weakest part of the framework since its results are more variable. Furthermore, we can also integrate the state-of-the-art LfD techniques to the planning module to allow the robots to perform more useful tasks, especially the ones that require precise skill such as “cutting” or “hammering”. However, in order to successfully perform these tasks, the robots also need to be equipped with the sufficient end-effector. From a vision point of view, although our

V2CNet can effectively translate videos to commands, the current network architecture considers each demonstration video equally and does not take into account the order of these videos. In real-world scenarios, the order of the actions also plays an important role to complete a task (e.g., "putting on sock" should happen before "wearing shoe"). Currently, we assume that the order of the sub-videos in a long manipulation task is known. Therefore, another interesting problem is to develop a network that can simultaneously segment a long video into short clips and generate a command sentence for each clip.

6. Conclusion

In this paper, we proposed V2CNet, a new deep learning framework to translate human demonstration videos to commands that can directly be used in robotic applications. Our V2CNet is composed of a classification branch to classify the fine-grained actions, and a translation branch to translate videos to commands. By jointly train both branches, the network can effectively encode both the spatial information in each frame and the temporal information across the time axis of the video. Furthermore, we also introduced a new large-scale dataset for the video-to-command task. The extensive experiments with different recurrent neural networks and feature extraction methods showed that our proposed method outperformed current state-of-the-art approaches by a substantial margin, while its output can be combined with the vision and planning modules in order to let the robots perform different tasks. Finally, we discussed the advantages and limitations of our approach in order to address the work for the future.

Acknowledgment

Anh Nguyen, Darwin G. Caldwell and Nikos G. Tsagarakis are supported by the European Unions Horizon 2020 research and innovation programme under grant agreement No 644839 (CENTAURO) and 644727 (CogIMon). Thanh-Toan Do and Ian Reid are supported by the Australian Research Council through the Australian Centre for Robotic Vision (CE140100016). Ian Reid is also supported by an ARC Laureate Fellowship (FL130100102). The authors would like to thank Luca Muratore for the help with the robotic experiments on WALK-MAN.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y and Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Akgun B, Cakmak M, Yoo JW and Thomaz AL (2012) Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In: *International Conference on Human-Robot Interaction (HRI)*.

Aksoy EE, Orhan A and Wörgötter F (2016) Semantic decomposition and recognition of long and complex manipulation action sequences. In: *International Journal of Computer Vision*.

Argall BD, Chernova S, Veloso M and Browning B (2009) A survey of robot learning from demonstration. *Robotics and Autonomous Systems* .

Calinon S, Evrard P, Gribovskaya E, Billard A and Kheddar A (2009) Learning collaborative manipulation tasks by demonstration using a haptic interface. In: *International Conference on Advanced Robotics (ICAR)*.

Cho K, van Merrienboer B, Gülçehre Ç, Bougares F, Schwenk H and Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078* .

Chrystopher L Nehaniv KD (2009) *Imitation and Social Learning in Robots, Humans and Animals Behavioural, Social and Communicative Dimensions*. Cambridge University Press.

Do TT, Nguyen A and Reid I (2018) Affordancenet: An end-to-end deep learning approach for object affordance detection. In: *International Conference Robotics and Automation (ICRA)*.

Donahue J, Hendricks LA, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K and Darrell T (2014) Long-term recurrent convolutional networks for visual recognition and description. In: *Computer Vision and Pattern Recognition (CVPR)*.

Finn C, Yu T, Zhang T, Abbeel P and Levine S (2017) One-shot visual imitation learning via meta-learning. In: *Conference on Robot Learning (CoRL)*.

Gan Z, Gan C, He X, Pu Y, Tran K, Gao J, Carin L and Deng L (2017) Semantic compositional networks for visual captioning. In: *International Conference on Computer Vision (CVPR)*.

Guadarrama S, Riano L, Golland D, Go D, Jia Y, Klein D, Abbeel P, Darrell T et al. (2013) Grounding spatial relations for human-robot interaction. In: *International Conference on Intelligent Robots and Systems (IROS)*.

He K, Zhang X, Ren S and Sun J (2016) Deep residual learning for image recognition. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R and Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kingma D and Ba J (2014) Adam: A method for stochastic optimization. In: *International Conference for Learning Representations (ICLR)*.

Koenemann J, Burget F and Bennewitz M (2014) Real-time imitation of human whole-body motions by humanoids. In: *International Conference on Robotics and Automation (ICRA)*.

Krizhevsky A, Sutskever I and Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems (NIPS)*.

Kruger V, Herzog DL, Baby S, Ude A and Kragic D (2010) Learning actions from observations. *Robotics & Automation Magazine* .

Kuehne H, Arslan A and Serre T (2014) The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lea C, Flynn MD, Vidal R, Reiter A and Hager GD (2017) Temporal convolutional networks for action segmentation and detection. In: *International Conference on Computer Vision (CVPR)*.

Lea C, Vidal R and Hager GD (2016) Learning convolutional action primitives for fine-grained action recognition. In: *International Conference Robotics and Automation (ICRA)*.

Lee K, Su Y, Kim TK and Demiris Y (2013) A syntactic approach to robot imitation learning using probabilistic activity grammars. *Robotics and Autonomous Systems* .

Mikolov T, Sutskever I, Chen K, Corrado GS and Dean J (2013) Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems (NIPS)*.

Muratore L, Laurenzi A, Mingo Hoffman E, Rocchi A, Caldwell DG and Tsagarakis NG (2017) Xbotcore: A real-time cross-robot software platform. In: *International Conference on Robotic Computing*.

Nguyen A, Do TT, Caldwell DG and Tsagarakis NG (2017a) Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. *arXiv preprint arXiv:1708.09011* .

Nguyen A, Do TT, Reid I, Caldwell DG and Tsagarakis NG (2018a) Object captioning and retrieval with natural language. *arXiv preprint arXiv:1803.06152* .

Nguyen A, Kanoulas D, Caldwell DG and Tsagarakis NG (2017b) Object-Based Affordances Detection with Convolutional Neural Networks and Dense Conditional Random Fields. In: *International Conference on Intelligent Robots and Systems (IROS)*.

Nguyen A, Kanoulas D, Muratore L, Caldwell DG and Tsagarakis NG (2018b) Translating videos to commands for robotic manipulation with deep recurrent neural networks. In: *International Conference Robotics and Automation (ICRA)*.

Pan Y, Yao T, Li H and Mei T (2017) Video captioning with transferred semantic attributes. In: *International Conference on Computer Vision (CVPR)*.

Pastor P, Hoffmann H, Asfour T and Schaal S (2009) Learning and generalization of motor skills by learning from demonstration. In: *International Conference on Robotics and Automation (ICRA)*.

Pastor P, Righetti L, Kalakrishnan M and Schaal S (2011) Online movement adaptation based on previous sensor experiences. In: *International Conference on Intelligent Robots and Systems (IROS)*.

Plappert M, Mandery C and Asfour T (2017) Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. *CoRR* abs/1705.06400.

Ramanishka V, Das A, Zhang J and Saenko K (2017) Top-down visual saliency guided by captions. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ramirez-Amaro K, Beetz M and Cheng G (2015) Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. *Artificial Intelligence* DOI:10.1016/j.artint.2015.08.009.

Rana MA, Mukadam M, Ahmadzadeh SR, Chernova S and Boots B (2017) Towards robust skill generalization: Unifying learning from demonstration and motion planning. In: *Conference on Robot Learning (CoRL)*.

Rhinehart N and Kitani KM (2017) First-person activity forecasting with online inverse reinforcement learning. In: *International Conference on Computer Vision (ICCV)*.

Rocchi A, Mingo Hoffman E, Caldwell D and Tsagarakis N (2015) OpenSoT: A Whole-Body Control Library for the Compliant Humanoid Robot COMAN. In: *International Conference on Robotics and Automation (ICRA)*.

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A and Fei-Fei L (2015) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* .

Simonyan K and Zisserman A (2014a) Two-stream convolutional networks for action recognition in videos. In: *Conference on Neural Information Processing Systems (NIPS)*.

Simonyan K and Zisserman A (2014b) Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556.

Stadie BC, Abbeel P and Sutskever I (2017) Third-person imitation learning. In: *International Conference on Learning Representations (ICLR)*.

Sun L, Jia K, Yeung DY and Shi BE (2015) Human action recognition using factorized spatio-temporal convolutional networks. In: *International Conference on Computer Vision (CVPR)*.

Sutskever I, Vinyals O and Le QV (2014) Sequence to sequence learning with neural networks. In: *Advances in Neural Information Processing Systems (NIPS)*.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J and Wojna Z (2016) Rethinking the inception architecture for computer vision. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tellex S, Kollar T, Dickerson S, Walter MR, Banerjee AG, Teller SJ and Roy N (2011) Understanding natural language commands for robotic navigation and mobile manipulation. In: *AAAI Conference on Artificial Intelligence*.

Toutanova K, Klein D, Manning CD and Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency network. In: *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

Tran D, Bourdev L, Fergus R, Torresani L and Paluri M (2015) Learning spatiotemporal features with 3d convolutional networks. In: *International Conference on Computer Vision (ICCV)*.

Tsagarakis NG, Caldwell DG, Negrello F, Choi W, Baccelliere L, Loc V, Noorden J, Muratore L, Margan A, Cardellino A, Natale L, Mingo Hoffman E, Dallali H, Kashiri N, Malzahn J, Lee J, Kryczka P, Kanoulas D, Garabini M, Catalano M, Ferrati M, Varricchio V, Pallottino L, Pavan C, Bicchi A, Settimi A, Rocchi A and Ajoudani A (2017) Walk-man: A high-performance humanoid platform for realistic environments. *Journal of Field Robotics* .

Venugopalan S, Rohrbach M, Donahue J, Mooney R, Darrell T and Saenko K (2016) Sequence to sequence - Video to text. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Venugopalan S, Xu H, Donahue J, Rohrbach M, Mooney R and Saenko K (2014) Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729* .

Welschehold T, Dornhege C and Burgard W (2016) Learning manipulation actions from human demonstrations. In: *International Conference on Intelligent Robots and Systems (IROS)*.

Xu J, Mei T, Yao T and Rui Y (2016) Msr-vtt: A large video description dataset for bridging video and language. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yang Y, Li Y, Fermuller C and Aloimonos Y (2015) Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. In: *AAAI Conference on Artificial Intelligence*.

Yao L, Torabi A, Cho K, Ballas N, Pal C, Larochelle H and Courville A (2015) Describing videos by exploiting temporal structure. In: *International Conference on Computer Vision (CVPR)*.

You Q, Jin H, Wang Z, Fang C and Luo J (2016) Image captioning with semantic attention. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu H, Wang J, Huang Z, Yang Y and Xu W (2015) Video paragraph captioning using hierarchical recurrent neural networks. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yu T, Finn C, Xie A, Dasari S, Zhang T, Abbeel P and Levine S (2018) One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557* .

Zhang T, McCarthy Z, Jow O, Lee D, Goldberg K and Abbeel P (2018) Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In: *International Conference on Robotics and Automation (ICRA)*.