Grounding Words in Visual Perceptions: Experiments in Spoken Language Acquisition

Fabio De Ponte

Final Project
MSc in Data Science and Artificial Intelligence
University of London

ABSTRACT

In recent years, Natural Language Processing models have shown compelling progress in generating and translating text. Yet, the symbols that are manipulated by these models are not inherent to the models themselves. The machine only calculates the probability that a specific token comes after another (or a group of others) and then produces a list of tokens, each of which has a certain probability to follow the previous one. There is no connection to sensory perceptions and the semantic interpretation of the outputs – as well as of the inputs – of these models is completely invisible to the system that produces them. Therefore, language cannot be used by the system to manipulate information about the perceived world. This is commonly referred to as the Symbol Grouding Problem and, as of today, there is not a generally accepted procedure to solve it. This project explores a possible solution, a sequence-to-sequence model trained over videos characterised by visual elements which reliably predict the presence of acoustic co-occurring elements. A dataset was created ad-hoc, and it includes 5 types of objects (namely pen, phone, spoon, knife and fork) and 5 actions (move to the left, to the right, up, down and rotate).

Two research questions were considered: whether such a model could map video features onto audio features, in fact producing a categorization without labels, where the categories would emerge from the parallel, simultaneous generalization of both input and target; and whether the model would be able to combine learned information about objects and movements to correctly describe a new combination, shown in a video it was not exposed to during training, a process that is defined as compositional semantics.

The experiment showed that the model was able to generalize simultaneously over videos and over the utterances that were paired with them. In fact, it produced sentences that were in some cases more accurate than the original ones, precisely because of the process of generalization. Once generalized, the co-occurring audio features paired with videos played the role of categories.

However, the results suggest also that the model did not develop the ability to combine information taken from different samples. It was trained over reduced datasets, where the videos depicting a specific object while moving were removed. Afterwards, the model was tested against those same videos. Through five rounds of training with different removed objects and 7,250 audio features produced, the correct combination never occurred. In other words, while symbol grounding seems to have been achieved, compositional semantics has not, as only if a combinatorial process can be applied, we can safely assume we left the domain of classification to enter that of symbol manipulation.

In conclusion, the experiment shows that sensory perceptions can be mapped onto one another with a sequence-to-sequence model trained over a dataset where elements coming from different sensory domains are paired. However, it is not sufficient to develop compositional semantics.

ACKNOWLEDGEMENT

This final project is the coronation of two years of efforts that were not only my own. I work full time as a journalist, a job that is often quite demanding. This MSc was for me an opportunity to finally achieve the technical skills I needed to develop artificial intelligence projects I have been thinking about for almost two decades. This would have not been possible without the sacrifice of my family. My partner Karolina and my daughter Marta often renounced on trips and time spent together to let me prepare my exams and coursework, and above all they endured my fervor of helpless enthusiasm at each new module.

I want to also acknowledge the tireless effort of my mother Annamaria to discuss motivations and objectives of this undertaking. As a former teacher, she is a source of continued inspiration, stimulating endless discussions about what the essence of learning is. And I can't help but thank my father, for all his work in our house and garden these past two years, which allowed me to focus on my studies.

Finally, I want to thank prof. Zimmer. His efforts to improve this MSc and his continuos commitment, along with his humor and passion for teaching, really made the difference for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTE

In the preparation of the present report, I made use of the couseworks I prepared for the Data Science Research Topics module. They were dedicated to the design of a project proposal and I used that work to start on this project.

1. INTRODUCTION

In recent years, Natural Language Processing models like GPT3 and BERT have shown compelling progress in interpreting and predicting text. Yet, while they help us understand the way languages work (offering at the same time valuable tools for different applications), they do not address the question of the relationship between words and sensory perceptions and of how language emerged in the first place. These questions date back centuries but have lately become known within the field of artificial intelligence as "the symbol grounding problem," since the issue was clearly defined by Harnad (1999), who stated it in these terms: "How can the semantic interpretation of a formal symbol system be made intrinsic to the system, rather than just parasitic on the meanings in our heads?"

Nowadays, even a model as simple as an n-gram conditional frequency model, trained over a sufficiently large corpus of text, can produce apparently meaningful text. However, the symbols that are manipulated by the system are not inherent to the system itself. The machine just calculates the probability that a certain token comes after another (or a group of others) and then produces a list of tokens, each of which has a certain probability to follow the previous one. In principle, this could be done by a human being on an unknown language. As John R. Searle (1980) famously argued in his paper "Minds, brains, and programs," a person could apply a similar method to a Chinese corpus and produce a Chinese text, without knowing the meaning of a single word.

Yet, recent advancements led a senior engineer at Google, Blake Lemoine, to claim that a language model, LaMDA, short for Language Model for Dialogue Applications, a Google's system for building chatbots, was "sentient" (Tiku, 2022). The company quickly dismissed the claim and, in a statement, Google spokesperson said: "There was no evidence that LaMDA was sentient (and lots of evidence against it)." In order to support his claim, Lemoine published a few astonishing fragments of chats between himself and the system. However, as surprising as they are, they do not seem to suggest the existence of a sentient mind behind. In fact, a simpler explanation is possible. Language models capture language structures and, with them, a fraction of the logic behind language itself. Mikolov, Yih and Zweig (2013) were able to design a model such that some seemingly logical operations between word vectors were possible, like "King - Man + Woman", resulting in a vector very close to "Queen", corresponding to the analogy "a man is to a king as a woman is to a queen." Operations like this between embedding vectors do not always work, even in more recent models. However, they show that sometimes it is possible to convert logical operations into geometrical operations. More generally, they suggest that human reasoning reflects in language and that a sufficiently complex language model can capture at least part of that logic. This could explain the surprising performances of the most advanced language models but leaves us with the unsolved problem of how a system can develop a meaningful semantic interpretation of the symbols that logic is applied to. Therefore, the challenge is to design a technique that would let the system define its own meaningful symbols. According to Harnad (1999), "there is only one viable route from sense to symbols: from the ground up." In other words, symbols have to be grounded to perceptions. There have been many attempts to do that and we will see some of them below.

2. BACKGROUND LITERATURE

Many attempts have been made at developing a method for grounding symbols to perceptions. One of the earliest was proposed by Roy (2000), who designed "a computational model which learns from untranscribed multisensory input" where "acquired words are represented in terms associations between acoustic and visual sensory experience." The model was designed to learn the same way children do, by discovering "words by searching for segments of speech which reliably predict the presence of visually co-occurring shapes." The author recorded a number of sessions of adults speaking to babies in a room. The adults were playing with the babies with toys, one toy at a time. The system included a speech processor that "converted spoken utterances into sequences of phoneme probabilities"; and a visual processor that "extracted statistical representations of shapes and color from images of objects." Phoneme probabilities and statistical representations of co-occurring images were stored in a short-term memory (STM) so that the model was able to predict the most probable next word, given an image. The experiment was a major step forward. However, it had an important drawback: not all utterances contained the name of the toy. Sometimes, adults said only, for example, "Here it comes!" referring to the toy car they were playing with. The research had two underlying aims: one was to build a model able to learn associations between co-occurring spoken words and visual elements; the other was to verify whether such a model could learn these associations even if not all audio recordings included the words to be learned, an ability that children seem to have. Roy performed the two steps simultaneously. Because of that it was impossible to discriminate which errors were caused by the inability of the model to learn words and which errors were instead caused by its unfitness to learn the same way children do. It would seem more reasonable to perform the two steps in sequence, not simultaneously.

Another interesting experiment was proposed by Tuci *et al.* (2011). The authors set up a virtual environment where subsequent generations of evolving robots were trained "to access linguistic instructions and to execute them by indicating, touching or moving specific target objects." In the course of the process, they were able to learn that the commands were composed of different parts. For example, "touch the pen" is made of two parts, "touch" and "the pen", while "move the spoon" is composed by "move" and "the spoon." Once they had learned that, they were able to generalize to new compositions, e.g. "touch the spoon," and execute them, even though they had never seen that specific command before. This, according to the authors, demonstrated "how the emergence of compositional semantics is affected by the presence of behavioural regularities in the execution of different actions." A drawback was that, while this experiment effectively demonstrated the principle, it dit not provide a method to let a system acquire language, other than within the very limited scopes of the experiment itself.

Another approach in the same direction was proposed by Suginta and Tani (2008). They focused on the creation of a geometrical n-dimensional space, where the geometric arrangements represented "the underlying combinatoriality" among symbols. In order to do it, they defined 26 actions and recorded 120 corresponding (algorithmically generated) sensor-motor time series for each and they were able to create a model where "the composition of symbols is realized by summing up their corresponding vectors," a compositional semantics potentially much more powerful than the one of Tuci *et al.* (2011). However, the method worked just for a few cases because the training data was not large enough.

Yet another approach came from Nolfi (2013). The author's goal was to "explain how simple communication forms emerged in the first place and how they evolved into structured communication systems." In order to do that, he set up an environment where a population of (physical) robots could evolve their behaviour. The robots had to find a specific area (called "food

area") of the environment and avoid another (called "poison area"). They were equipped with a two-colors led that – at the beginning – emitted a random light. Four different series of experiments were ran, varying the fitness function. After a certain number of generations, the robots learned that when more than one light was concentrated in a certain area, it was likely that it was the food area and that they should move in that direction. Moreover, with certain fitness functions that fostered the group coordination, they learned to emit a specific color when they were close to the food area and another when they were close to the poison area: what at the beginning was only a random emission of signals developed in a stable communication system, with robots able to both communicate and interpret the signs of others. These results suggest that "an expansion of the individuals' behavioral skills might have been one of the main factors that triggered the origins of language."

More recently, a number of other approaches were proposed. One came from Gonzalez-Billandon *et al.* (2019), who converted "the audio signal to an embedding space where embeddings for the same words are closer than different words, regardless of speaker." In order to achieve this, they used a Vector Quantized-Variational Autoencoder (VQ-VAE) network. Then, the embeddings were associated to images. The project is still ongoing and the results have not been released yet.

Another method came from Wang *et al.* (2020), who proposed MAXSAT, a SATNet layer that can be integrated into neural network architectures that could successfully be used to learn logical structures. Their technique was subsequently used by Topan, Rolnick and Si (2021), to map visual inputs to symbolic variables without explicit supervision, through a self-supervised pre-training pipeline.

Another step forward was made possible by Shao et al. (2021), whose research was based on a *learning from demonstration* method. With a model that partly resembled the one of Tuci et al. (2011), they leveraged the "something something" video database, that offers images of thousands of actions on objects, paired with a linguistic label, composed by a description of the action in the form "do <something> on <something>" (hence the name). They designed a system based on reinforcement learning. Linguistic and visual elements produced an action and the feedback was based on the visual inspection of the results of the action, much like a baby learns how to grab a fork thanks to the example and the encouragements of a parent, who gives a positive or negative feedback. Although the experiment was succesful, much like in the case of the work of Tuci et al. (2011), it dit not propose a general procedure to let a system acquire language.

A different approach was tried by Tan and Bansal (2019), who proposed LXMERT (Learning Cross-Modality Encoder Representations from Transformers), a large-scale transformer model. In essence, it is a framework designed to learn direct vision-to-language connections and it represents a step forward in the task of describing the content of images in words. It outperforms previous models on visual question-answering datasets. However, it does not include sound or any perception other than vision. Therefore, it does not offer a direct contribution to tackle the symbol grounding problem and it does not address the question of compositional semantics.

A proposal in this direction was instead put forward by Liu, Li and Cheng (2021), who replaced text with audio. They proposed a new general-purpose neural sound synthesis (V2RA) network, based on generative adversarial networks (GANs), that was able to generate sound directly from visual inputs. In their work, the task was "formulated as a regression problem to map a sequence of video frames to a sequence of raw audio waveform." The problem was solved "with three main ingredients, namely a video encoder […], a V2RA-GAN […] and an audio optimization" unit. This model paves the way for the design of a general method to associate visual to acoustic features and

vice versa. It represents a significant step in the direction of the solution of the symbol grounding problem. However, it was designed to reproduce any kind of sound, and the authors had in mind mostly noise produced by specific objects (e.g. cars, motorcycles, scrolling water, etc.). It was not designed for spoken language and it does not address the question of compositional semantics.

3. METHODOLOGY

3.1 Objectives of the research

The project focused on the following research objectives:

1. Verify whether it is possible for a system to learn words from an unlabeled dataset of spoken utterances and visual representations, through a sequence-to-sequence neural network.
2. Verify if, once such words are learned, compositional semantic would emerge, i.e. if the system would be able to combine them to compose sentences that were not present in the training dataset.

The project built on Roy (2000) and Tuci *et al.* (2011) works:

1. Roy's work was expanded in two directions:
   a) **Dataset.** To train his model, Roy created a quite complex dataset, composed by audio-recordings of adults talking to a child while playing with a toy, and images of the toy in question shot from different angles and perspectives. However, the name of the toy often did not occur in the speech. Therefore, I created another dataset of videos, where each audio recording includes the word to be learned.
   b) **Code.** Roy developed a rather complex system to extract features from audio and video, through speech and visual processors, and built another system to manage Short Time Memory and Long Time Memory. The speech processor converted "spoken utterances into sequences of phoneme probabilities" and, at a rate of 100Hz, it computed "the probability that the past 20 milliseconds of speech belonged to each of 39 English phoneme categories or silence," while the visual processor generated "second order statistics." Color was represented by computing "a two-dimensional histogram of illumination-normalized RGB pixel values from the area of the image occupied by the target object." I built a simpler model, leveraging Keras libraries. I extracted features making use of pre-trained convolutional neural networks and replaced the Short Time Memory and Long Time Memory modules with a sequence-to-sequence network composed of LSTM layers.

2. Following Tuci *et al.* (2011), I also tried to achieve compositional semantics. In order to do it, videos showing objects both moving and staying still were included in the dataset. During training, the model was exposed to videos showing, for example, still forks and moving spoons. Then it was tested on videos showing moving forks to verify if it could combine the information received and produce a new sentence.

3.2 Dataset

There are many available datasets online that at first sight could serve the purposes of this project. However, at a closer scrutiny, none seems to have the needed characteristics. One of the candidates was the "something something" video database. It is a collection of more than 100,000 videos showing actions performed on objects. Each video is paired with a description of the action in the form "do <something> on <something>." The drawback with this dataset is that the decriptions are written and not spoken. In order to make that dataset suitable for the symbol grounding, the captions should be converted to spoken recordings. If we did that, it would make for a perfect dataset to address the symbol grounding along with the compositional semantics problems.

Another possible choice was ImageNet. It was not suitable for the project for the same reason, that is the lack of audio recordings. Also, in the case of ImageNet there are images and not videos, so it would be difficult to introduce actions into the model.

Yet another possible candidate was the COCO dataset (short for "Common Objects in Context"). It includes pictures labeled with multiple object dectections and image segmentations that could allow for the achievement of compositional semantics. Yet again, it lacks audio recordings, therefore the model would end up managing externally imposed labels, rather than discovering by itself cross-sensory maps and using those as symbols for compositional semantics.

Other datasets were taken into account, but none seemed to be suitable for this project. Therefore, I decided to create my own dataset. It is composed of 1,000 videos, each showing an object – either staying still or moving – while a voice reads a sentence, for example "this is a pen," that refers to what we see. There are twenty voices in total: ten voices are natural – i.e. they have been recorded by real people – while the other ten are artificial. The ten people gave their permission to the publication of the recordings. They are three males and seven females, aged between 10 and 62 years old. The artificial voices were produced through the services of the website www.murf.ai (I possess the commercial rights, they were released against payment).

3.2.1 The structure

Each video is exactly 3 seconds long. It is 180x180 pixel sized and the audio is sampled to 16 KHZ. Size and sample rate are just above the line that makes video and audio intelligible. The length allows for a movement to be shown and still offers room for some forward and backward shift in time for data augmentation purposes. The dataset is published online and it is released under licence Creative Commons Attribution-ShareAlike 3.0 Generic (CC BY-SA 3.0). Details can be found in appendix A.

Each voice recorded 50 sentences: the first 25 described still objects, the other 25 referred to moving objects. A corresponding video was recorded for each sentence. Audio and video were then paired and saved into a single MP4 file, that was trimmed to a standard length of 3 seconds. It is a balanced dataset: each object and each action is represented 5 times for each group of 50 videos.

Number	Spoken utterance	Video content	Times
1-5	"This is a **pen**"	A pen on a table	5
6-10	"This is a **phone**"	A phone on a table	5
11-15	"This is a **spoon**"	A spoon on a table	5
16-20	"This is a **knife**"	A knife on a table	5
21-25	"This is a **fork**"	A fork on a table	5

Table 1. Objects: 25 audios and 25 videos. In the first column, the position of the sample in the group. In the second column, the content of the spoken sentences. In the third column, the content shown within the video. In the fourth column, the number of times videos with such contents were shot and included in the dataset (for each voice).

Number	Spoken utterance	Video content	Times
26	"**Move the** pen **to the left**"	A pen moving to the left	Once
27	"**Move the** pen **to the right**"	A pen moving to the right	Once
28	"**Move the** pen **up**"	A pen moving up	Once
29	"**Move the** pen **down**"	A pen moving down	Once
30	"**Rotate the** pen"	A pen rotating	Once
31	"**Move the** phone **to the left**"	A phone moving to the left	Once
32	"**Move the** phone **to the right**"	A phone moving to the right	Once
33	"**Move the** phone **up**"	A phone moving up	Once
34	"**Move the** phone **down**"	A phone moving down	Once
35	"**Rotate the** phone"	A phone rotating	Once
36	"**Move the** spoon **to the left**"	A spoon moving to the left	Once
37	"**Move the** spoon **to the right**"	A spoon moving to the right	Once
38	"**Move the** spoon **up**"	A spoon moving up	Once
39	"**Move the** spoon **down**"	A spoon moving down	Once
40	"**Rotate the** spoon"	A spoon rotating	Once
41	"**Move the** knife **to the left**"	A knife moving to the left	Once
42	"**Move the** knife **to the right**"	A knife moving to the right	Once
43	"**Move the** knife **up**"	A knife moving up	Once
44	"**Move the** knife **down**"	A knife moving down	Once
45	"**Rotate the** knife"	A knife rotating	Once
46	"**Move the** fork **to the left**"	A fork moving to the left	Once
47	"**Move the** fork **to the right**"	A fork moving to the right	Once
48	"**Move the** fork **up**"	A fork moving up	Once
49	"**Move the** fork **down**"	A fork moving down	Once
50	"**Rotate the** fork"	A fork rotating	Once

Table 2. Actions: 25 audios and 25 videos. In the first column, the position of the sample in the group. In the second column, the content of the spoken sentences. In the third column, the content shown within the video. In the fourth column, the number of times videos with such contents were shot and included in the dataset (for each voice).

3.2.2 Artificial voices

The choice of adopting artificial voices raised a problem: there were groups of five identical utterances among the first 25 on the list. While people never read a sentence in exactly the same way, artificial systems do. Therefore, the dataset had groups of identical voice recordings. In order to solve the issue, a preliminary data augmentation was implemented: for each group of five identical voices, the first was left unchanged, the volume of the second was lowered, the volume of third was raised, the fourth was delayed by 0.5 seconds and the fifth was delayed by 1 second. Here are the details of the four operations:

- Decrease volume: multiply volume by a factor 0.5 (applied to voices 2, 7, 12, 17 and 22)
- Increase volume: multiply volume by a factor 1.5 (applied to voices 3, 8, 13, 18 and 23)
- Delay by 0.5: add delay of 0.5 seconds (applied to voices 4, 9, 14, 19 and 24)
- Delay by 1.0: add delay of 1 second (applied to voices 5, 10, 15, 20 and 25)

Artificial voices from 26 to 50 were left unchanged. All natural voices were instead normalized. Once all this was completed, I proceeded to the general data augmentation process, both on audio and video.

3.2.3 Data augmentation: new sets

The videos were augmented by flipping image, increasing and decreasing bright, increasing saturation and zooming in. Audios were augmented as well, increasing and decreasing speed, increasing and decreasing pitch and increasing volume. Lastly, data was split into training, validation and test sets.

Five operations were applied to audios, another five to videos. All were performed through Ffmpeg and Rubberband utilities. Here is the detailed list:

Audio

1. Decrease speed: multiply by a factor 0.7 (through Ffmpeg utility)
2. Increase speed: multiply by a factor 1.3 (through Ffmpeg utility)
3. Increase pitch: multiply by a factor 2 (through Rubberband utility)
4. Increase pitch: multiply by a factor 10 (through Rubberband utility)
5. Increase volume: multiply by a factor 2 (through Ffmpeg utility)

Video (all performed through Ffmeg utility)

6. Flip:
    1. horizontally 1-25 and 28, 29, 30, 33, 34, 35, 38, 39, 40, 43, 44, 45, 48, 49, 50 (move up, move down and rotate)
    2. vertically 26, 27, 31, 32, 36, 37, 41, 42, 46, 47 (move to the right and to the left)
7. Increase brightness: add a factor 0.2
8. Decrease brightness: subtract a factor 0.1
9. Increase saturation: multiply by a factor 2
10. Zoom: zoom in (and crop accordingly) by a factor 1.2

A complete scheme of the data augmentation can be found in appendix A.

3.2.4 Data augmentation: pairing sets

After completing the data augmentation, each of the five augmented audio sets of 1,000 samples (plus the original one) was paired with each of the five augmented video sets of 1,000 samples (plus the original one). In total, we got 36 datasets (36,000 samples). Here is the scheme of the pairing:

	0 - Original	**6** - Flip	**7** - Light +	**8** - Light -	**9** - Saturation	**10** - Zoom
0 - Original	0-0	0-6	0-7	0-8	0-9	0-10
1 - Speed -	1-0	1-6	1-7	1-8	1-9	1-10
2 - Speed +	2-0	2-6	2-7	2-8	2-9	2-10
3 - Pitch +	3-0	3-6	3-7	3-8	3-9	3-10
4 - Pitch ++	4-0	4-6	4-7	4-8	4-9	4-10
5 - Volume +	5-0	5-6	5-7	5-8	5-9	5-10

Table 3. Paring sets.

3.3 Network architecture

Once the dataset was ready, a system was designed to map visual elements to spoken utterances. Video files were processed by two pre-trained neural networks, namely Wav2vec and CLIP, that extracted respectively acoustic and visual features. Then a sequence-to-sequence neural network mapped the extracted features of the visual part onto the extracted features of the acoustic part.

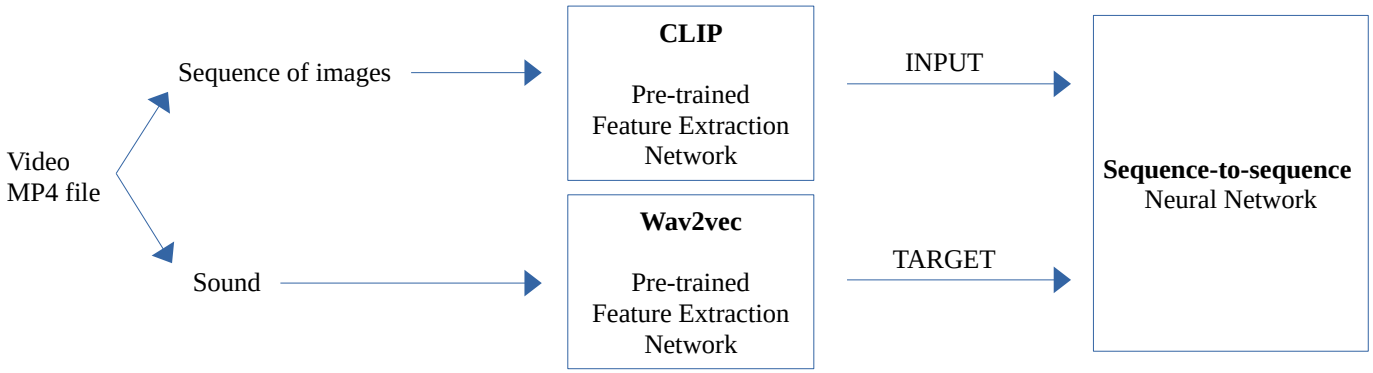

Fig. 1. The structure of the model

3.3.1 The audio features extractor

The audio features extractor leveraged Wav2vec, a model introduced by Baevski *et al.* (2020) at Facebook for self-supervised learning of representations from raw audio. This library fits the needs of this project in three ways:

1. It returns features vectors of 150 integer values, a manageable size that makes the sequence-to-sequence model trainable on a relatively small machine like the one I had available, a PC with an Intel i7 CPU and an Nvidia RTX3080 GPU.
2. It returns features that can be in turn decoded to a written text by the same library. This makes the evaluation of the predictions of the model easier, as it allows the conversion of the predicted features into text and the comparison with the expected ones.

3. The features can be padded with zeros up to any length, so they can be easily converted into vectors for neural networks training.

As we had a dataset composed of 36,000 videos, we got an audio features matrix sized 36,000 x 150. The code of the audio features extractor, a modified version of a program published online by Subramanian (2021), is available in appendix B.

3.3.2 Audio features discarded

As a preliminary check, audio features were decoded to written text, through the function *batch_decode* (see appendix B) provided by the library Wav2vec. From this verification, it emerged that not all features produced intelligible text. In particular, the sentences recorded by real people contained many errors. The reason is probabily that, as they are non-native speakers, their pronunciation was not good enough for the library.

	UTTERANCE		UTTERANCE
0	THES IS A PEN	25	MOVE THE PANAR TO THE LEFT
1	DITIS A PEN	26	MOVE THE PAMER TO THE RIGHT
2	BIZIS A PEN	27	MOVE THE PEN UP
3	BIZIS A PEN	28	MOVE I THE PEND DOWN
4	DITISA PEN	29	NOTE TO THE PEM
5	THESE IS A FORM	30	MOVE DE FORM T O THE LEFT
6	TESISIPONA	31	MOVE THE FALLER TO THE RIGHT
7	THESE IS TE FORNER	32	MOVE DE FONAP
8	THES IS A FORMNER	33	MOVE THE FORN DOWN
9	THES IS A FONE	34	ROCEDE DE FORME
10	THESE IS A SPOON	35	MOVED DISPOON TO THE LEFT
11	THESE IS A SPOON	36	MO VI THE SPOON TO THE RIGHT
12	DISISAS POON	37	MOVE DESPOON UP
13	DISESPUNE	38	MOVED A SPOON DOWN
14	THES IS A SPOON	39	ROTE TO THE SCOON
15	THES IS A KNIFE	40	MOVE TE NIGHT TO LEFT
16	THESE IS A KNIVE	41	MOVED A KNIFE TO THE RIGHT
17	THESE IS A KNIVE	42	MOVE THE KNIFE UP
18	DES IS A LIFE	43	MOVED ANIFE DOWN
19	THESE IS A KNIFE	44	THEY WERE TA TO DENIVE
20	DES IS A FORK	45	MOVED THE FORG TO THE LEFT
21	DES IS A FORK	46	MOVE THE FORK TO THE RIGHT
22	DISIS E FORK	47	MOVE THE FORK UP
23	BISIS A FORK	48	MOVE THE FORK DOWN
24	THES IS THE FORK	49	ROTATE DE FORC

Table 4. Real voices utterances converted to text by Wav2vec.

Moreover, the artificial voices that had their pitch multiplied by a factor of ten for data augmentation purposes resulted in unintelligible decoded text as well. Therefore, all real voices and a part of the artificial ones were discarded, along with the corresponding videos. The dataset was thus reduced in size and as a result was composed by only 14,500 samples.

	UTTERANCE		UTTERANCE
0	RULIPAN	25	MARKET A PAN OT WOCK
1	WOBUN	26	WORK THE PIL OF AROLP
2	MOVE UPON	27	LO THE BARK
3	MOVE UPON	28	UPON ARN
4	MOAPUN	29	AND E TOLD YOU A PLUN
5	LEVEN A FON	30	WHERE COULD I FIR TACK
6	WO YO BEFON HIM	31	MOTEN GET FOND OF IT GET OUT
7	MAKI AFRON YOU	32	MORABLE FORNACK
8	NETIT AFIRMME	33	YOURBE GET FOANY GAWY
9	RAVE TE TORIA	34	RIGHTALL TO THE FERN
10	AMO LEF CAROINGON	35	MARGE AS PERIT LOT
11	AMO LEF CAROINGON	36	MORE CUL THE FREE GET IT OUT
12	NE LEFT BREWING ON	37	NOW IT US FERY HAP
13	WO LEF PERLON	38	O YO FOM FARN
14	BELSPERIN	39	I TOLD HIM IS PURNY
15	MONOW	40	NOT LAD COME UP
16	THESE IS A KNIVE	41	ON LOUT ROUT
17	WO ROUT	42	WHAT THE MOW
18	BO NOW	43	WORK WILE WOW
19	WE LITTE ON NO	44	OR A CULTIBLE MOUSE
20	MEE FARK	45	WON BEFORE I COULD UP
21	BEE E FICK	46	WON'T GET FORRD FOR YOU TOGA
22	BE ITTE FRANK	47	MOVE THE FORK UP
23	DI HE FACK	48	MOVE THE FORK DOWN
24	MEE TEFRIC	49	ROTATE DE FORC

Table 5. Artificial voices utterances augmented through pitch multiplication by 10 and converted to text by Wav2vec.

It should be noted that this does not necessarily mean that the discarded samples are useless. They could be used in a model that does not make use of Wav2vec. In fact, whether the features are intelligible or not for a speech-to-text converter is not relevant for symbol grounding, as long as similar things are told consistently in similar ways.

However, because we needed to convert the features to text in order to evaluate the predictions of the model, we had to discard elements that could not be interpreted by Wav2vec. In other words, a choice had to be made between keeping those samples and keeping the Wav2vec library.

3.3.3 The video features extractor

For the video features extractor, we made use of the CLIP model developed by Radford *et al.* (2021) at OpenAI. It is suitable for our purposes because it returns a vector of 512 float values. Again, this is a manageable size. However, it does not provide a tool to convert the features back to a video or to any other directly readable form. For this reason, while Wav2vec makes it relatively simple to evaluate the predictions of a video-to-audio neural network, CLIP does not do the same for an audio-to-video network. Therefore, we focused on the transformation of video features into audio features and not the opposite, developing a sequence-to-sequence video-to-audio network.

The code of the video features extractor is a modified version of an implementation of CLIP presented online by Iashin (2021) and it is available in appendix C.

For completeness, it is worth mentioning that models other than CLIP were considered. In particular, I3D and RAFT were applied to extract features, but they were less appropriate:

- I3D returns float vectors of length 2,048. The first half, 1,024 values, named in the model the "RGB" part, captures colors and shapes. The second half, named in the model the "FLOW" part, captures action. It seemed very suitable for our purposes. However, it allowed only a lower performance.
- RAFT (short for Recurrent All-Pairs Field Transforms for Optical Flow) returns a matrix of 2x2x180x180 per sample, i.e. four frames of the video. It is too large to be managed, as 1,000 samples produce a features files of 1GB. For 14,500 samples, it would produce a file of 14.5GB.

3.3.4 The sequence-to-sequence model

Once we got the two features matrices, it was necessary to develop a code to map the video features onto the audio features. It seems that there is no consolidated approach to this task, but sequence-to-sequence models appear to be among the most suitable options. They are generally employed for text-generation purposes and I made use of a code developed by Chollet (2017), that was originally written as a translation model. This kind of model is suitable for our purpose because it maps a list of one-hot encoded integers (generated through a dictionary from the sentence to be translated) to another list of one-hot encoded integers (generated through another dictionary).

The sequence-to-sequence models are composed of two parts: an encoder and a decoder. Each of them includes one or more LSTM layers. During training, the encoder basically compresses the sequence into a vector, while the decoder learns a series of conditional frequency distributions: for each value and each vector coming from the encoder, it calculates the most likely value to come next. In terms of translation process, this means that the decoder calculates the most probable next word given the last predicted word and the vector coming from the encoder, which summarizes information about the whole sentence to be translated.

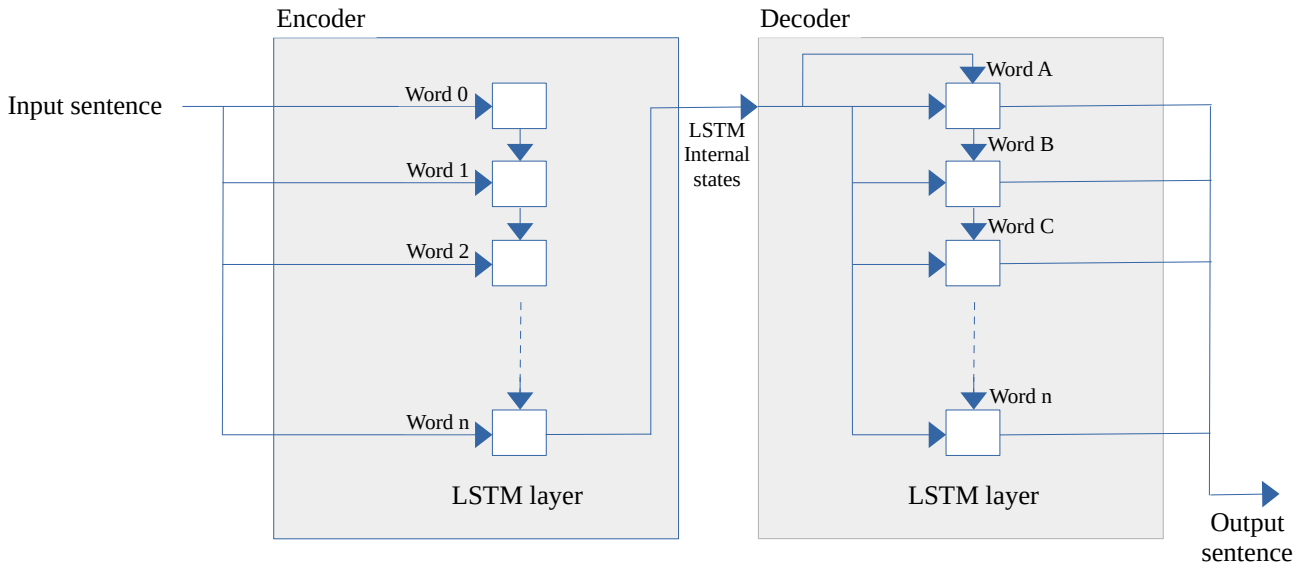

Fig. 2. A sequence-to-sequence model

In our case, we can consider each word of the input sentence one value of the 512-long video features vector and each word of the target translated sentence a value of the 150-long audio features target vector. In fact, mapping video features onto audio features may be interpreted as a translation task: on the input side, we have a sequence that represents the contents of a video; on the output side, we expect a sequence that represents the same contents expressed in audio format.

In terms of code, the implemented model (see appendix D) was composed overall by three layers. The first was the encoder LSTM layer, which received the video one-hot encoded features. It returned an output and two internal states. The output was discarded, while a vector composed by the two internal states combined was passed onto the next LSTM layer, the decoder one. The output of this second layer was then passed onto a dense layer, with activation function softmax, so that the output of the model was again a one-hot encoded vector.

One of the problems to be solved was that, while the audio features produced from Wav2vec are integers (ranging between 0 and 28), and thus fit for one-hot encoding, the video features produced by CLIP model are float. Therefore, they were normalized between 0 and 1, then multiplied by 100 and finally rounded to the integer. This way, they could be one-hot encoded in vectors of size 101 (values ranging from 0 to 100).

3.3.5 Metrics

In order to identify the best performing model, different configurations were tried. In particular, the size of the layers, the learining rate, the optmizer and the directionality (mono or dual) were modified. Three metrics were adopted to evaluate performances:

1. The loss on training and test datasets. The activation of the last layer was Softmax and the loss function was categorical cross entropy. Therefore, the loss was a measure of the error on the prediction of the one-hot vectors. Accuracy, precision and recall were also calculated.
2. The cosine distances between the predicted features vectors and the expected features vectors.
3. The visual comparison of the predicted with expected texts.

3.3.6 The compositional semantics experiment

As we said, while the first aim was to design a model that would be able to map videos onto audios, the second was to verify wether such a model could develop compositional semantics. In order to do that, an experiment was set up. The videos containing a specific moving object were removed from the dataset, while videos showing the same object laying still were kept, along with all other objects, both moving and still. The model was trained on the reduced dataset, and it was tested against the removed videos. The audio features predicted by the model were converted into text and so it was possible to compare them with the intended ones.

For example, at one point all videos of moving pens were removed, so that the model was trained only on videos showing pens laying still (paired with the utterance “this is a pen” read by different voices) and other objects (spoons, forks, knives and phones) both staying still (paired with utterances like “this is a spoon”) and moving (paired with utterances like “move the spoon to the right”). No videos showing a pen moving to the right and the utterance “move the pen to the right”

were included in the training set. The experiment consisted in verifying whether the model was able to compose the utterance “move the the pen to the right,” once such a video was given to the network. In order to do that, the model should have been able to combine the information coming from the shape of the pen and the information coming from the movement of other objects. I repeated the experiment five times, with one object at a time.

3.3.7 Limitations

It can be argued that the use of pre-trained neural networks for features extraction within this project is a contradiction, because they were trained on labeled datasets. Therefore, if the aim is to build a system that is able to manipulate information without labels, we cannot include such networks. This is a reasonable objection, but we could avoid the use of pre-trained networks only by making use of large datasets and very deep models. The present project is a middle-step in that direction. Once there is a working framework, we will be able to get rid of the pre-trained networks, and, following Liu, Li and Cheng (2021), build a network able to generate sounds directly from visual inputs and vice versa.

4. RESULTS AND DISCUSSION

4.1 Features pre-processing

As we saw earlier, audio features of each sample consist of 150 integer values, ranging from 0 to 28. In order to prepare them for the sequence-to-sequence model, each value was converted to a one-hot vector of length 29.
Video features returned by the CLIP model are vectors of 512 floats. As they could not be directly one-hot encoded, they were as we said normalized between 0 and 1, then multiplied by 100 and finally rounded to integers. This way, they could be converted to one-hot vectors of length 101 (ranging from 0 to 100).

4.2 Symbol grounding: identifying the best performing model

4.2.1 Loss

Twenty different configurations of the sequence-to-sequence model were tried, modifying the size of the layers, the learining rate, the optmizer and the directionality. The results can been seen in table 6.

Model	Layers	Video One-hot size	Bidir.	Drop.	Batch	Opt.	L.R.	Train loss	Train acc.	Val. loss	Val. acc.	Test loss	Test acc.	Train cosine	Test cosine
C_01	256 + 256	101	No	0	64	Rmsp.	0.001	0.0253	0.9904	0.0475	0.9832	0.0463	0.9833	57.51	58.97
C_02	512 + 512	101	No	0	64	Rmsp.	0.001	0.0139	0.9938	0.0405	0.9870	0.0392	0.9874	52.36	55.88
C_03	1024 + 1024	101	No	0	64	Rmsp.	0.001	0.0125	0.9941	0.0365	0.9883	0.0352	0.9886	44.27	54.74
C_04	2048 + 2048	101	No	0	64	Rmsp.	0.001	0.0133	0.9938	0.0352	0.9881	0.0335	0.9884	51.16	56.44
C_05	1024 + 1024	101	No	0	64	Sgd	0.01	0.5466	0.8386	0.5446	0.8398	0.5467	0.8400	52.11	51.83
C_06	1024 + 1024	101	No	0	64	Adam	0.001	0.0115	0.9944	0.0339	0.9886	0.0336	0.9890	36.65	50.12
C_07	1024 + 1024	101	No	0	64	Sgd	0.001	0.7007	0.8236	0.6983	0.8241	0.6988	0.8244	52.11	51.83
C_08	1024 + 1024	101	No	0	64	Adam	0.01	0.4960	0.8497	0.4946	0.8499	0.4962	0.8494	52.25	51.71
C_09	1024 + 1024	201	No	0	64	Adam	0.001	0.0121	0.9943	0.0335	0.9886	0.0330	0.9890	39.08	52.80
C_10	1024 + 1024	71	No	0	64	Adam	0.001	0.0334	0.9873	0.0456	0.9836	0.0453	0.9837	43.03	55.20
C_11	64 + 128	101	Yes	0	64	Adam	0.001	0.0841	0.9680	0.1002	0.9620	0.0998	0.9620	60.87	58.97
C_12	1024 + 2048	101	Yes	0	64	Adam	0.001	0.0538	0.9797	0.0571	0.9792	0.0563	0.9792	63.91	62.73
C_13	512 + 1024	101	Yes	0	64	Adam	0.001	0.0229	0.9911	0.0453	0.9838	0.0464	0.9836	62.26	54.06
C_14	256 + 512	101	Yes	0	64	Adam	0.001	0.0125	0.9943	0.0373	0.9875	0.0358	0.9881	43.08	51.93
C_15	128 + 256	101	Yes	0	64	Adam	0.001	0.0244	0.9910	0.0465	0.9833	0.0458	0.9835	56.90	56.88
C_16	1024 + 1024	101	No	0.2 1st layer	64	Adam	0.001	0.0333	0.9873	0.0366	0.9864	0.0358	0.9867	48.18	53.37
C_17	1024 + 1024	101	No	0.2 2nd layer	64	Adam	0.001	0.0784	0.9712	0.0647	0.9760	0.0643	0.9763	65.67	64.50
C_18	1024 + 1024	101	No	0.2 both layers	64	Adam	0.001	0.043	0.9831	0.0352	0.9863	0.0350	0.9863	49.89	54.57
C_19	1024 + 1024	101	No	0	128	Adam	0.001	0.0126	0.9943	0.0369	0.9876	0.0370	0.9878	34.18	50.52
C_20	1024 + 1024	101	No	0	**32**	Adam	0.001	0.0113	0.9945	0.0327	0.9894	0.0319	0.9894	35.88	54.29

Table 6. Summary of the performances of the 20 models evaluated.

First, a network with layers composed of 256 neurons was tried, for 100 epochs with optimiser RMSprop and learning rate 0.001. In table 6, this model is indicated as C_01. The train loss was 0.025 and the test loss was 0.046. Then, a larger network, C_02, was tried (two 512-sized layers). It performed better: train and test loss were respectively 0.014 and 0.039.

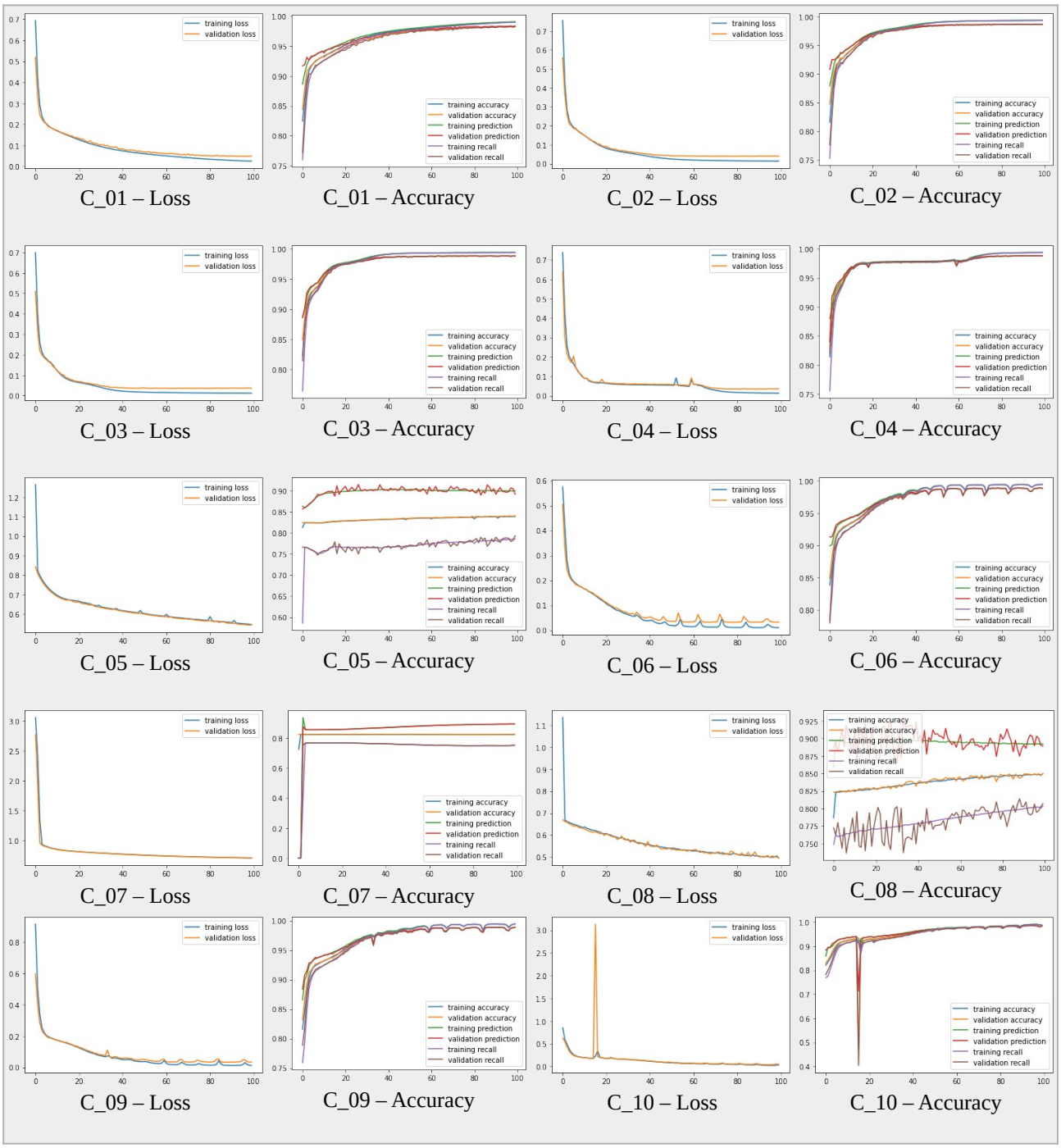

Fig. 3a. Loss and accuracy of the models 1-10. Apart from accuracy, precision and recall are also shown.

The next model, C_03, was even larger (1024+1024) and resulted in 0.012 and 0.035. Given the apparent progress, an even larger network was tested: 2048+2048. This time, the performance on the training set declined to 0.013, while it improved on the test set: 0.033. Therefore, we set the size

at 1024+1024 and, with C_05, we tried stochastic gradient descent, with learning rate 0.01. The results were poor as the loss skyrocketed for both training and test sets to 0.547. With model C_07, a lower learning rate was tried, 0.001, but the results remained disappointing. The next attempt, C_06, was with Adam optimizer. In this case, the results were encouraging: 0.012 and a 0.037. With the next model, C_08, a learning rate of 0.01 was tried. It did not work well, both the train and the test loss resulted 0.496.

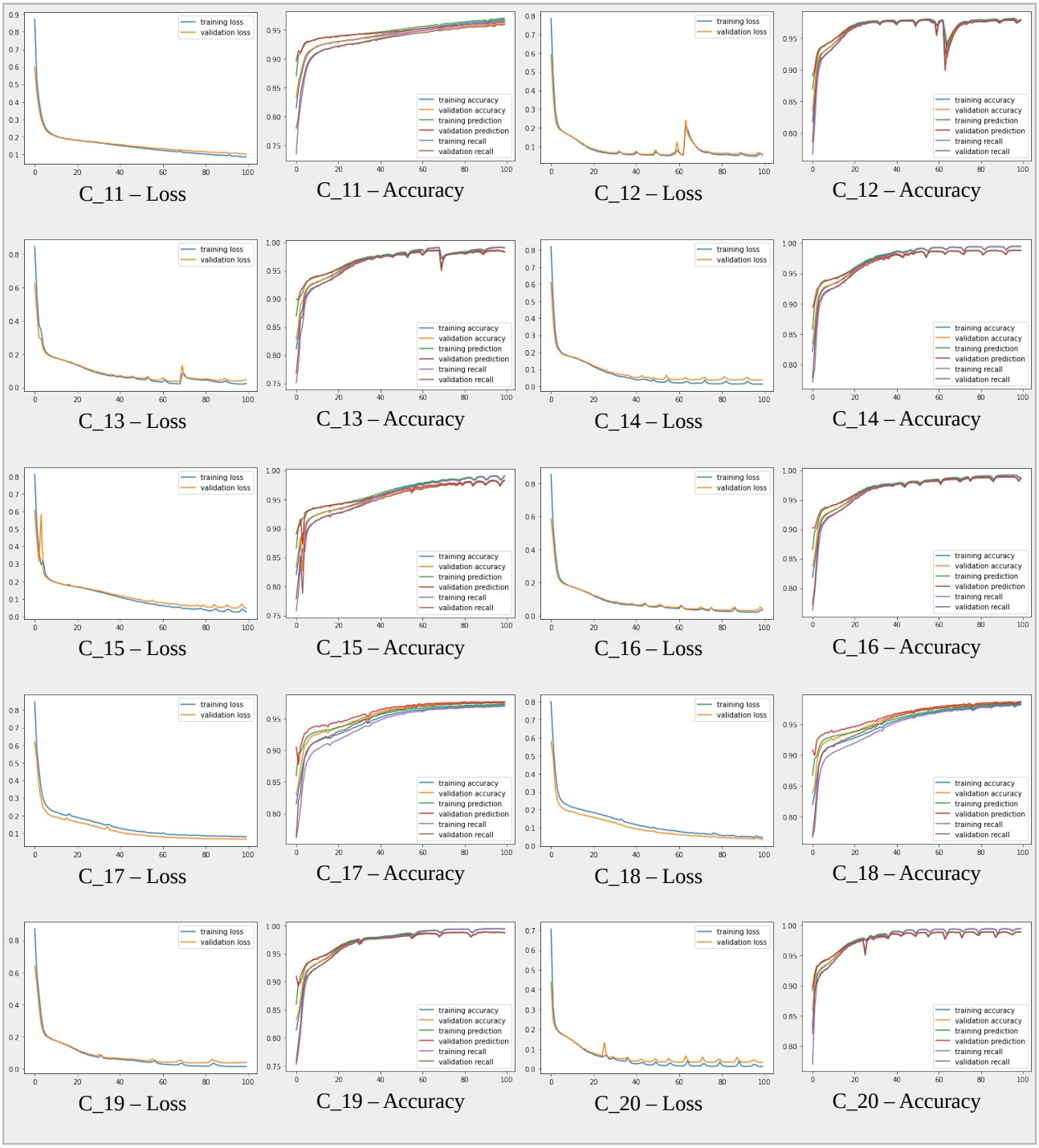

Fig. 3b. Loss and accuracy of the models 11-20. Apart from accuracy, precision and recall are also shown.

So far, the best model seemed to be C_06. A possible bottleneck was the one-hot encoding of the video features. The 512 values were normalized to the range 0-1, multiplied by 100 and rounded to integer. Therefore, each value was one-hot encoded in a vector sized 101. In order to double the precision, the normalized values were multiplied by 200 and then rounded, so that the one-hot vectors were of size 201. Model C_09 was trained with these vectors, but it did not show a better performance: training loss was 0.012, test loss was 0.033, the first slightly higher and the second slightly lower than C_06's losses. Overall, it did not bring much difference. With C_10, the opposite was tried: the model was trained with 71-sized one-hot input vectors. This time, the performance decreased: train loss was 0.033 and test loss was 0.045.

Another possible improvement could be to make use of bidirectional LSTM layers. These were introduced with model C_11, with encoder layer sized 64 and decoder layer sized 128. The performance was quite poor: train loss 0.084 and test loss 0.100. With C_12, the same was tried with layers sized 1024 and 2048, which resulted in train and test loss respectively 0.054 and 0.056. With C_13, C_14 and C_15, three middle-sized models were tried: 512-1024, 256-512 and 128-256. Among the models with bidirectional layers, the best performing one resulted C_14, with training loss 0.013 and test loss 0.036. These results were almost the same of C_06, but the model was more computationally expensive.

As we can see from the graphs, model C_06 did not seem to overfit significantly, as train and validation curves appeared very close. However, in order to verify if the gap could be completely closed, dropout was applied with model C_16, in the measure of 0.2 on the encoder LSTM. The performance decreased significantly: train loss was 0.033, nearly three times the train loss of C_06, while test loss was 0.036, slightly higher than C_06's. A second experiment was carried with C_17, where a 0.2 droupout was applied on the decoder LSTM. Again, the performance degraded to 0.078 and 0.064. For completeness, the dropout was also tried on both layers and, unsurprisingly, the result was still poor: 0.043 and 0.035. After having left aside the dropout, some tests were made on the batch size. It was first increased to 128, resulting in a train loss of 0.013 and test loss of 0.037, almost the same as C_6. Then, it was decreased to 32, and the result was a train loss of 0.011 and a test loss of 0.032. It was a slight improvement on the performance of C_6, but it made the training process much slower.

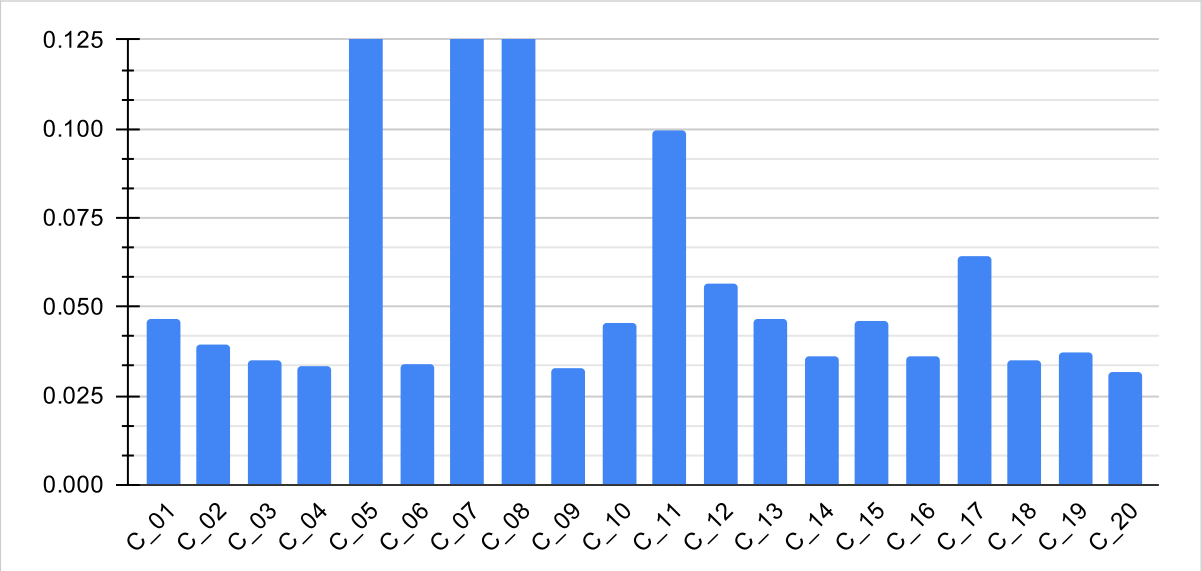

Fig. 4. Test loss. C_06, C_09 and C_20 show the lowest test loss. Note that C_05, C_07 and C_08 present a test loss much higher than 0.125, it was cut here for visualization purposes.

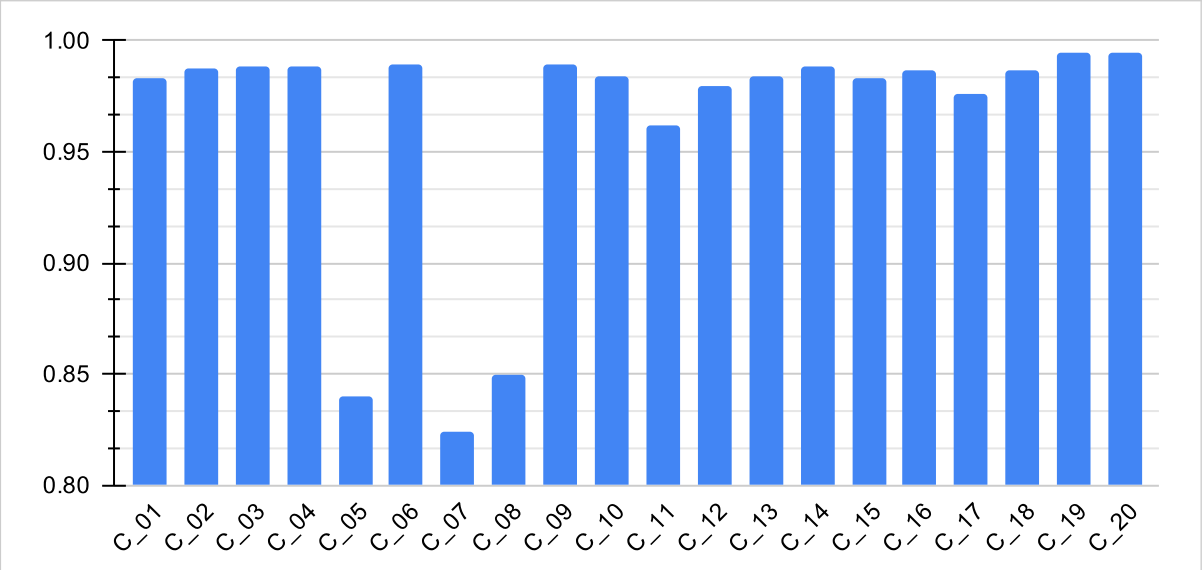

Fig. 5. Test accuracy. C_06, C_09 and C_20 show the highest accuracy. On the contrary, C_05, C_07 and C_08 show the lowest accuracy.

4.2.2. Cosine distance

Along with loss, another performance metric was adopted: the cosine distance between the predicted features and the expected ones. This measure was calculated comparing predicted and target vectors on 100 samples extracted from the train set and 100 from the test set. As we can see in table 6, these measures mostly confirmed the observations that could be done through the loss curves. In particular, the model C_06 showed the lowest average test cosine distance, which was 50.12.

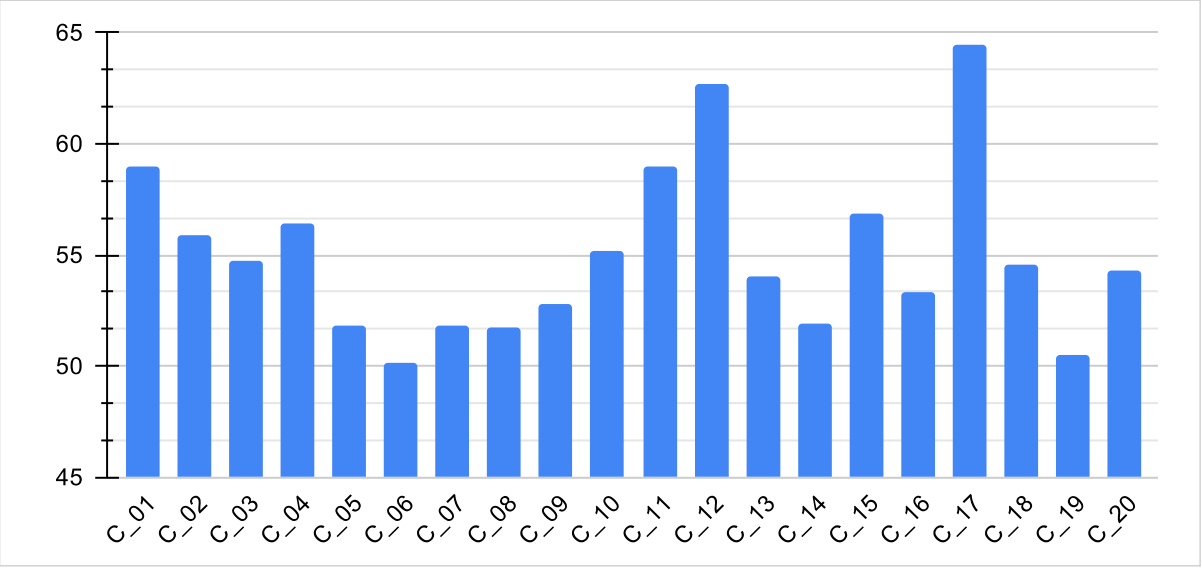

Fig. 6. Test cosine distance. C_06 shows the lowest value.

4.2.3. Visual inspection

Another important mean of evaluation was the visual inspection. The model C_06 produced better formed sentences, that were also more similar to the expected ones. We can see below a sample of 11 randomly picked sentences, as predicted by the models C_01, C_06, C_11, C_12 and C_14. The left column shows the sentence originally paired with the video (as converted by Wav2vec from the original audio into text), the right column shows the predicted sentence. We can clearly see that C_06 produces the best predictions:

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
0	MOVE THE FORK TO THE RIGHT	MOVE THE FORK TO THE RIGHT
1	ROTATE THE KNIFE	WROT TAKE THE PHINE
2	THIS IS A SPOON	THIS IS A PEN
3	MOVE THE FORK TO THE RIGHT	MOVE THE FORK TO THE LEFT
4	MOVE THE SPOON DOWN	MOVE THE SPOON TO THE LEFT
5	HIS IS A SPURN	THIS IS A PHONE
6	MOVED THE FORK TO THE LEFT	MOVE THE PEN TO THE LEFT
7	ROTATE THE SPOON	RO TAKE THE FORK
8	LUSA PHONAP	MOVE THE PHONE TO THE RIGHT
9	THIS IS A SPOON	THIS IS A SPOON
10	MOVE THE PHONE TO THE RIGHT	MOVE THE PEN UP

Table 7. Model C_01. Original and predicted utterances

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
0	MOVE THE FORK TO THE RIGHT	MOVE THE FORK TO THE RIGHT
1	ROTATE THE KNIFE	WROTATE THE KNIFE
2	THIS IS A SPOON	THIS IS A SPOON
3	MOVE THE FORK TO THE RIGHT	MOVE THE FORK TO THE RIGHT
4	MOVE THE SPOON DOWN	MOVE THE SPOON DOWN
5	HIS IS A SPURN	THIS IS A SPERN
6	MOVED THE FORK TO THE LEFT	MOVE THE FORK TO THE LEFT
7	ROTATE THE SPOON	ROTATE THE SPOON
8	LUSA PHONAP	MOVES THE PHONE UP
9	THIS IS A SPOON	THIS IS A SPOON
10	MOVE THE PHONE TO THE RIGHT	MOVE THE PHONE TO THE RIGHT

Table 8. Model C_06. Original and predicted utterances

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
0	MOVE THE FORK TO THE RIGHT	THIS IS A FORK
1	ROTATE THE KNIFE	OVE THE KNIFE O
2	THIS IS A SPOON	THIS IS A SPOON
3	MOVE THE FORK TO THE RIGHT	THIS IS A FORK
4	MOVE THE SPOON DOWN	OOV THE SPONE UP
5	HIS IS A SPURN	THIS IS A SPOON
6	MOVED THE FORK TO THE LEFT	OVED THE SPOON TO THE LEFT
7	ROTATE THE SPOON	THIS IS A SPOON
8	LUSA PHONAP	THIS IS A PHONE
9	THIS IS A SPOON	THIS IS A SPOON
10	MOVE THE PHONE TO THE RIGHT	MOVE THE PHONE DOWN

Table 9. Model C_11. Original and predicted utterances

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
0	MOVE THE FORK TO THE RIGHT	THIS IS A KNIFE
1	ROTATE THE KNIFE	THIS IS A KNIFE
2	THIS IS A SPOON	THIS IS A KNIFE
3	MOVE THE FORK TO THE RIGHT	THIS IS A FOWN
4	MOVE THE SPOON DOWN	IS IS A SPOON
5	HIS IS A SPURN	THIS IS A SPOON
6	MOVED THE FORK TO THE LEFT	THIS IS A FORK
7	ROTATE THE SPOON	THIS IS A FOWN
8	LUSA PHONAP	THIS IS A FORK
9	THIS IS A SPOON	THIS IS A SPOON
10	MOVE THE PHONE TO THE RIGHT	THIS IS A KNIFE

Table 10. Model C_12. Original and predicted utterances

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
0	MOVE THE FORK TO THE RIGHT	MOVE THE FORK TO THE RIGHT
1	ROTATE THE KNIFE	WROTATE THE KNIFE
2	THIS IS A SPOON	THIS IS A SPOON
3	MOVE THE FORK TO THE RIGHT	MOVE THE FORK TO THE RIGHT
4	MOVE THE SPOON DOWN	MOVE THE SPOON DOWN
5	HIS IS A SPURN	THIS IS A SPERN
6	MOVED THE FORK TO THE LEFT	MOVE THE FORECAP
7	ROTATE THE SPOON	ROTATE THE SPOON
8	LUSA PHONAP	THIS IS A PONE
9	THIS IS A SPOON	THIS IS A SPOON
10	MOVE THE PHONE TO THE RIGHT	MOVED THE PHUNE DOWN

Table 11. Model C_14. Original and predicted utterances

As we can see in table 8, the model C_06 was able to produce sentences that in some cases were even more more understandable than the original. In the batch of 11 sentences that we extracted from the test dataset:

- It correctly predicted “move the fork to the right” (sample number 0), “this is a spoon” (n. 2), “move the fork to the right” (n. 3), “move the spoon down” (n. 4), “rotate the spoon” (n. 7), “this is a spoon” (n. 9) and “move the phone to the right” (n.10).
- It predicted “wrotate the knife” (n. 1), a substantially correct output with the minor error of adding a “W” to the word “rotate”.
- It predicted “This is a spern” (n. 5) for the video of a spoon staying still that was originally paired with a scarsely comprehensible utterance, namely “his is a spurn”, instead of “this is a spoon”. In fact, generalizing over similar videos and over the utterances that were paired with them, the model showed a result that turned out to be better than the original. This was again the case with the following utterance: the model returned features corresponding to “move the fork to the left” (n. 6) whereas the original video had been paired with “moved the fork to the left”, with a “d” at the end of “move” that made the utterance slightly incorrect. And again (n. 8) the model returned “moves the phone up”, which was a strong improvement over the utterance originally paired with the video, that was interpreted by the Wav2vec library as “lusa phonap”.

Having performed these tests, C_06 was picked as the best perfoming model and it was therefore the model adopted for the next phase, the compositional semantics test.

4.3. Compositional semantics test

Once video features were mapped onto audio features, the experiment presented in paragraph 3.3.6 was performed in order to verify if the model developed the ability of combine elements autonomously, i.e. if it developed compositional semantics capabilities.
In order to do that, five subsequent tests were performed, one for each object: pen, phone, spoon, knife and fork. For each of them, a reduced dataset was prepared, removing the videos that showed the object moving to the left, to the right, up, down and rotating. The model was trained on the reduced dataset, resulting in 13,050 samples, and then tested against the 1,450 videos that had been removed. The results are shown in table 12.

	TRAIN AND TEST ON REDUCED DATASET				**TEST ON MOVING OBJECT VIDEOS**		
OBJECT	**Train loss**	**Train accuracy**	**Test loss**	**Test accuracy**	**Loss**	**Accuracy**	**Cosine distance**
PEN	0.02	0.99	0.04	0.99	0.73	0.88	59.81
FORK	0.01	0.99	0.03	0.99	0.8	0.88	66.28
PHONE	0.03	0.99	0.04	0.99	0.73	0.89	59.59
KNIFE	0.03	0.99	0.04	0.98	0.86	0.87	64.59
SPOON	0.02	0.99	0.04	0.99	0.97	0.87	61.47

Table 12. Model C_06 trained on reduced dataset and tested on removed moving object.

As we can see, loss incresed dramatically when the model was tested over videos of moving objects that it was not exposed to during training. We can clearly see that it is not a generalization problem, because the *Test loss* column shows that, when tested against videos previously unseen but belonging to known groups, the model showed a loss ranging between 0.030 and 0.043. Yet, when exposed to new kinds of videos, its loss skyrocketed, ranging between 0.73 and 0.97. The model was able to generalize when similar videos were present during training. However, it was not able to combine information from the videos that showed the objects staying still and information about the movement applied to other objects, to form a sentence composed by “move” and the name of the object.

	ORIGINAL UTTERANCE	**PREDICTED UTTERANCE**
1	OVED THE PAN TO THE LEFT	THIS IS A PEN
2	MOVE THE PEN TO THE RIGHT	THIS IS A PEN
3	MOVE THE PEN UP	THIS IS A PEN
4	OVE THE PEN DOWN	THIS IS A PEN
5	ROTAKE THE PEN	THIS IS A PEN
6	MOVE THE PEN TO THE LEFT	OVE THE IFE TO THE LEFT
7	MOVE THE PEN TO THE RIGHT	THIS IS A PEN
8	MOVE THE PENNOP	MOVE THE KNIFE DOWN
9	MOVE THE PEN DOWN	THIS IS A PEN
10	ROTATHE THE PEN	THIS IS A PEN

Table 13. Model C_06 tested on PEN videos removed from dataset during training. On the left, the utterances originally associated with the videos of the moving object removed from the training dataset. On the right, the prediction of the model.

In each table, 10 predictions per object (out of 1,450) are shown. As we can see in table 13, the model did not predict a sentence containing either “move the pen” or “rotate the pen” when exposed to a moving pen. Nor was it able (table 14) to predict “move the phone” or “rotate the phone” when exposed to the moving phone videos. Similarly, it could not predict any of the correct sentences for

the other three objects. In the dataset of 14,500 videos, each object was represented 2,900 times: 1,450 times laying still and 1,450 moving. The model was trained five times, each time leaving aside the videos of a particular object depicted while moving. Each time it was tested against those videos. In total, it returned 7,250 audio features, converted then to text. It cannot be shown here due to space contraints, but a through inspection showed that the combination never occurred.

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
1	MOVED THE PHONE TO THE LEFT	THIS IS A PHAWN
2	MOVED THE PHONE TO THE RIGHT	MOTHE THE SPOON TO THE LEFT
3	OOV THE PHON UP	IS IS A FARN
4	OVE THE FALM DOWN	THIS IS A PHONE
5	ROUTAKE THE PHONE	THIS IS A PHAWN
6	MOVE THE PHONE TO THE LEFT	THIS IS A PHONE
7	MOVE THE PHONE TO THE RIGHT	THIS IS A PHONE
8	MOVE THE PHONA	THIS IS A PHONE
9	MOVE THE FONE DOWN	THIS IS A FOWN
10	ROTATE THE PHONE	THIS IS A PEN

Table 14. Model C_06 tested on PHONE videos removed from dataset during training. On the left, the utterances originally associated with the video of the moving object removed from the training dataset. On the right, the prediction for that video.

The model mostly favoured the shape of the object, predicting a sentence in the form “this is...,” thus ignoring the movement. In a minority of cases, it recognised the movement, but it failed to combine the information with the shape of the object and simply predicted the utterance “move” accompanied by a different object, often the most similar. For example, “move the pen” was frequently mistaken for “move the knife.”

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
1	OOVED THE SPOON TO THE LEFT	IS IS A SPOON
2	MOVED THE SPOON TO THE RIGHT	IS IS A SPOON
3	MOVE THE SPOON UP	THIS IS A SPOON
4	OOVE THE SPOON DOWN	THIS IS A PHON
5	RO TAKE THE SPOON	THIS IS A SPOON
6	MOVE THE SPOON TO THE LEFT	THIS IS A KNIFE
7	MOVE THE SPOON TO THE RIGHT	THIS IS A KS OI
8	MOVE THE SPOONER	MOVE THE PAN TO THE RIGHT
9	MOVE THE SPOON DOWN	THIS IS A PEN
10	ROTATE THE SPOON	MOVE THE KNIFE TO THE LEFT

Table 15. Model C_06 tested on SPOON videos removed from dataset during training. On the left, the utterances originally associated with the video of the moving object removed from the training dataset. On the right, the prediction for that video.

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
1	MOVE THE KNIFE TO THE LEFT	THIS IS A KNIFE
2	MOVE THE KNIFE TO THE RIGHT	THIS IS A KNIFE
3	OO THE KNIFE UP	THIS IS A KNIFE
4	OE THE KNIFE DOWN	THIS IS A KNIFE
5	ROTATE THE KNIFE	THIS IS A PKNIFE
6	MOVE THE KNIFE TO THE LEFT	MOVE THE PENNOB
7	MOVE THE KNIFE TO THE RIGHT	MOVE THE PEN DOWN
8	MOVE THE KNIFE	MOVE THE PEN DOWN
9	MOVE THE KNIFE DOWN	THIS IS A PEN
10	ROTATE THE KNIFE	THIS IS A KNIFE

Table 16. Model C_06 tested on KNIFE videos removed from dataset during training. On the left, the utterances originally associated with the video of the moving object removed from the training dataset. On the right, the prediction for that video.

	ORIGINAL UTTERANCE	PREDICTED UTTERANCE
1	LOVE THE FORK TO THE LEFT	THIS IS A FORK
2	MOVE THE FORK TO THE RIGHT	OVE THE KNIFE UP
3	OO THE FORECUP	THIS IS A FORK
4	MOVE THE FORK DOWN	THIS IS A FORK
5	ROU TAKE THE FORK	OVES THE PEN TO THE RATE
6	MOVE THE FORK TO THE LEFT	THIS IS A FORK
7	MOVE THE FORK TO THE RIGHT	THIS IS A FORK
8	MOVE THE FORCA	THIS IS A FORK
9	MOVE THE FORK DOWN	THIS IS A FORK
10	ROTATE THE FORK	THIS IS A FORK

Table 17. Model C_06 tested on FORK videos removed from dataset during training. On the left, the utterances originally associated with the video of the moving object removed from the training dataset. On the right, the prediction for that video.

5. CONCLUSIONS

5.1 Symbol grounding and compositional semantics

This project had two aims. The first was to design a system (composed, as we saw, of three neural networks) that had to be able to directly map, without the intervention of labels, videos onto spoken utterances. In other words, the objective was for the system to be able to directly produce a correct sentence when exposed to a video, learning words directly from co-occurrence of visual and acoustic elements. This way, instead of being externally imposed, words would have been extracted directly from sensory perceptions, a practice that in literature is defined symbol grounding. Ideally, the system should have returned spoken utterances but, for reasons of evaluation, the predicted audio features were converted to text.

The second aim was to verify whether, once the mapping was done, such a system would develop the ability to compose sentences that were not present in the training dataset, combining information gathered from different samples. This is called compositional semantics.

The results shown above suggest that the ability to directly produce a correct sentence when exposed to a video has been achieved. The model was able to generalize on both sides of the dataset and map videos onto audios without labels.

The other aim of the project was to achieve compositional semantics. The results suggest that the model did not develop the ability to combine information taken from different samples. It was trained over reduced datasets, where the videos depicting a specific object while moving were removed. Afterwards, the model was tested against those same videos. The research hypothesis was that the model could develop the ability to produce a sentence that decribed the action and the object at the same time, even though that particular combination was not present in the training dataset. The hypothesis was not verified. Through five rounds of training with different datasets, and 7,250 audio features produced, that combination never occurred.

With this experiment, we tried to capture the power of categorization carried by words into an artificial system. In fact, the co-occurring audio features associated with each video, once generalized, play the role of categories. However, they are categories that the model itself extracts from the sensory perceptions through a process of generalization and not externally given labels. The fact that words are intimately linked to categories is not surprising, as with language comes indeed “the ability to generalize,” as Oliver Sacks (1990, p.42) pointed out in “Seeing voices”, a book devoted to the relationships between sensory perceptions and language. As Lupyan (2005) argues, “merely perceiving an object does not require categorizing it. In contrast, naming an object (whether to communicate to another individual or for your own benefit) does require placing it into a category.” However, classification into categories is not enough. Once things have names, the necessity of a grammar arises, in order to allow the acquired categories to be manipulated. And, as Corballis (2002, p.37) illustrates, the origin of grammar could very well have been compositional semantics:

> The simplest events consist of objects and actions, such as *baby screams*, *snake approaches*, or *apple falls*. Suppose, for example, that an animal’s experience includes five meaningful objects and five meaningful actions. If each object is associated with a single action, so that only babies scream or only apples fall, then there are only five events to be signaled, and five *event* symbols will do the trick; the objects do not need to be distinguished from the actions associated with them. But if all possible combinations of

objects and actions can occur, then it would be more economical to learn five symbols for the objects and five for the actions, making ten in all, than to learn twenty-five symbols to cover all their possible combinations. This might be the source of protolanguage, leading eventually to grammar.

5.2 Further work

This experiment shows that it is possible to achieve symbol grounding through a sequence-to-sequence model trained over a dataset with audio and video co-occurring elements. It seems safe to assume that the approach could be scaled up with a larger dataset and a deeper network.

Moreover, the work could be expanded in several directions:

1. Apply an attention layer introduced by Bahdanau *et al.* (2015) to the sequence-to-sequence model. It is possible that, though it was not achieved with the simple model adopted, compositional semantics could be developed once the model includes an attention mechanism. Moreover, it is also possible that a model based on transformers could show the ability of combining information from different samples.
2. Give up pre-trained features extraction networks. For real symbol grounding to be fully achieved, the system should not include networks that were trained over labels. They could be replaced by autoencoders, a solution that could present two benefits: it could offer the possibility of reconstructing the original audio and video files, something that could in turn allow a performance evaluation of an audio-to-video model and thus open the possibility in that direction; moreover, the layers of the autoencoders could be directly integrated within the sequence-to-sequence neural network, something that could simplify the model.
3. Expand the dataset. The dataset developed for this project was relatively small, with one thousand original videos and only five objects represented. It could be expanded, with the addition of more objects, either staying still or moving, and new voices.
4. Apply the model to the “something something” dataset. That dataset seems perfectly suitable for purposes of compositional semantics. However, it lacks audio. In order to overcome the problem, captions – at least a fraction of them – could be converted to spoken utterances through artificial voices.

WORD COUNT: 7,784
(excluding cover page, table of contents,
tables, figures, references and appendices)

REFERENCES

Anderson, P., Wu, Q., Teney D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., van den Hengel, A. (2018), *Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments,* in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018, pp. 3674–3683. Available at: https://openaccess.thecvf.com/content_cvpr_2018/html/Anderson_Vision-and-Language_Navigation_Interpreting_CVPR_2018_paper.html (Accessed: 16 September 2022)

Baevski, A., Zhou, H., Mohamed, A.R. and Auli, M. (2020), *Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*, in 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada. Available at: https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf (Accessed: 14 September 2022)

Bahdanau, D., Cho, K., and Bengio, Y. (2015), *Neural Machine Translation by Jointly Learning to Align and Translate*, in 3rd International Conference on Learning Representations 2015. Available at: https://arxiv.org/pdf/1409.0473.pdf (Accessed: 6 September 2022)

Cangelosi, A., Greco, A. & Harnad., S (2000) *From robotic toil to symbolic theft: Grounding transfer from entry-level to higher-level categories.* Connection Science, Vol.12, No. 2, 2000, pp. 143-162. Available at: https://doi.org/10.1080/09540090050129763 (Accessed: 8 May 2022)

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724-1734. Available at: https://aclanthology.org/D14-1179 (Accessed: 19 September 2022)

Chollet, F. (2017), Character-level recurrent sequence-to-sequence model, *Keras*, 29 September. Available at: https://keras.io/examples/nlp/lstm_seq2seq (Accessed: 5 September 2022)

Corballis, M.C. (2002), *From hand to mouth. The origins of language*, Princeton and Oxford, Princeton University Press.

Fanello S.R., Ciliberto C., Natale L., Metta G. (2013) *Weakly supervised strategies for natural object recognition in robotics* in IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, May 6-10, 2013. Available at: https://ieeexplore.ieee.org/document/6631174 (Accessed: 23 May 2022)

Gehring, J., Auli, M., Grangier, D., Yarats, D. and Dauphin, Y.N. (2017), *Convolutional Sequence to Sequence Learning*, in Proceedings of the 34th International Conference on Machine Learning - Volume 70 - August 2017 Pages 1243–1252. Available at: https://arxiv.org/abs/1705.03122 (Accessed at: 6 September 2022)

Gentner, D., (1981) *Some interesting differences between verbs and nouns,* in Cognition and Brain Theory, 1981, vol. 4, pp. 161-178. Available at: https://groups.psych.northwestern.edu/gentner/papers/Gentner81c.pdf (Accessed: 22 May 2022)

Gonzalez-Billandon, J., Grasse, L., Sciutti, A., Tata, M. and Rea, F. (2019), *Cognitive Architecture for Joint Attentional Learning of word-object mapping with a Humanoid Robot*, in IEEE/RSJ International Conference on Intelligent Robots and Systems 2019. Available at: https://www.researchgate.net/publication/344432053_Cognitive_Architecture_for_Joint_Attentional_Learning_of_word-object_mapping_with_a_Humanoid_Robot (Accessed: 12 June 2022)

Hannagan, T., Agrawal, A., Cohen, L. and Dehaene S. (2021), *Emergence of a compositional neural code for written words: Recycling of a convolutional neural network for reading*, in Pnas, Vol. 118 - No. 46. Available at: https://www.pnas.org/doi/10.1073/pnas.2104779118 (Accessed: 6 September 2022)

Harnad, S. (1990), *The Symbol Grounding Problem. Physica D 42: 335-346.* Available at: https://arxiv.org/abs/cs/9906002v1 (Accessed: 7 May 2022)

Harnad, S. (2002), *Symbol grounding and the origin of language*, University of Southampton Institutional Research Repository. Available at: *https://eprints.soton.ac.uk/256471* (Accessed: 16 September 2022).

Huang, H., Wong, R.K. (2021), *Attention-based Seq2seq Regularisation for Relation Extraction*, in 2021 International Joint Conference on Neural Networks (IJCNN). Available at: https://ieeexplore.ieee.org/abstract/document/9533807 (Accessed: 6 September 2022)

Iashin, V. (2021), 'CLIP', *Video Features Documentation*. Available at: https://iashin.ai/video_features/models/clip (Accessed: 5 September 2022)

Liu, S., Li, S. and Cheng, H. (2021), *Towards an End-to-End Visual-to-Raw-Audio Generation with GAN*, in IEEE Transaction on Circuits and Systems for Video Technology, Vol. 32, Issue: 3. Available at: https://ieeexplore.ieee.org/document/9430540 (Accessed: 6 September 2022)

Luong, M.T., Pham, H., and Manning, C.D. (2015), E*ffective Approaches to Attention-based Neural Machine Translation*, in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Available at: https://arxiv.org/abs/1508.04025 (Accessed at: 6 September 2022)

Lupyan, G. (2005) *Carving nature at its joints and carving joints into nature: how labels augment category representations*, Progress in Neural Processing, Vol. 16, 2005, pp. 87-96. Available at: https://doi.org/10.1142/9789812701886_0008 (Accessed: 8 May 2022)

Nolfi S. (2013) "Emergence of communication and language in evolving robots" in Lefebvre, C., Comrie, B. and Cohen H., *New Perspectives on the Origins of Language*, 2013, pp. 533–554. Available at: https://doi.org/10.1075/slcs.144.20nol (Accessed: 8 May 2022)

Pasquale, G., Ciliberto, C., Odone, F., Rosasco, L. and Natale, L. (2019), *Are we done with object recognition? The iCub robot's perspective*, in Robotics and Autonomous Systems, Volume 112, February 2019, Pages 260-281. Available at: https://www.sciencedirect.com/science/article/pii/S0921889018300332 (Accessed: 12 June 2022)

Pinker, S. (1996), *Language Learnability and Language Development (1984/1996)*. Cambridge, MA: Harvard University Press.

Pinker, S.(1994) How could a child use verb syntax to learn verb semantics?, in Lingua Vol. 92, April 1994, Pages 377-410. Available at: https://www.sciencedirect.com/science/article/pii/0024384194903476 (Accessed 22 Maj 2022)

Prickett, B., Traylor, A., and Pater, J. (2018), *Seq2Seq Models with Dropout can Learn Generalizable Reduplication*, in Proceedings of the 15th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 93–100 Brussels, Belgium, October 31, 2018. Available at: https://aclanthology.org/W18-5810.pdf (Accessed: 5 September 2022)

Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, A., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. and Sutskever, I. (2021), *Learning Transferable Visual Models From Natural Language Supervision*, in Proceedings of the 38 th International Conference on Machine Learning, PMLR 139, 2021. Available at: http://proceedings.mlr.press/v139/radford21a/radford21a.pdf (Accessed: 14 September 2022).

Ramanathan, V., Tang, K., Mori, G. and Fei-Fei, L. (2015), *Learning Temporal Embeddings for Complex Video Analysis* in Proceedings of International Conference on Computer Vision (ICCV) 2015. Available at: https://arxiv.org/abs/1505.00315 (Accessed 26 June 2022)

Roger, E., Banjac, S., Thiebaut de Schotten, M. and Baciua, M. (2022), *Missing links: The functional unification of language and memory* in Neuroscience & Biobehavioral Reviews, Volume 133, February 2022. Available at: https://www.sciencedirect.com/science/article/pii/S0149763421005601 (Accessed 26 June 2022)

Roy, D. (2000), *Grounded speech communication*, in Proceedings of the 6th International Conference on Spoken Language (ICSLP 2000), vol. 4, pp. 69-72. Available at: https://www.isca-speech.org/archive_v0/archive_papers/icslp_2000/i00_4069.pdf (Accessed: 22 May 2022)

Roy, D. (2002), *Learning Visually Grounded Words and Syntax of Natural Spoken Language*, Computer Speech & Language, Volume 16, Issues 3–4, July–October 2002, pp. 353-385. Available at: https://www.media.mit.edu/cogmac/publications/evol_comm_2002.pdf (Accessed: 12 May 2022)

Roy, D. (2003), *Grounded Spoken Language acquisition*: Experiments in Word Learning, in IEEE Transactions on Multimedia, Vol. 5, No. 2, June 2003. Available at: https://www.media.mit.edu/cogmac/publications/ieee_multimedia_2003.pdf (Accessed: 12 May 2022)

Roy, D. (2005), *Grounding words in perception and action: computational insights*, in Trends in Cognitive Sciences Vol.9 No.8, August 2005. Available at: https://lsm.media.mit.edu/papers/Roy_TICS_2005.pdf (Accessed: 22 May 2022)

Sacks, O. (2012), *Seeing voices*. 3rd edn. London: Picador books.

Searle, J.R. (1980), *Minds, brains, and programs*, in The behavioral and brain sciences, pp. 417-457. Available at: https://www.law.upenn.edu/live/files/3413-searle-j-minds-brains-and-programs-1980pdf (Accessed: 12 May 2022)

Selsam, D., Lamm, M., Bunz, B., Liang, P., de Moura, L., and Dill, D. L. (2018), *Learning a sat solver from single-bit supervision*, in Proceedings of The International Conference on Learning Representations (ICLR) 2019. Available at: https://arxiv.org/abs/1802.03685 (Accessed: 12 June 2022)

Shao, L., Migimatsu, T., Zhang, Q., Yang, K. and Bohg, J. (2021), *Concept2Robot: Learning manipulation concepts from instructions and human demonstrations*, in The International Journal of Robotics Research, Vol. 40, Issue 12-14, Oct. 2021. Available at: https://journals.sagepub.com/doi/abs/10.1177/02783649211046285 (Accessed: 16 September 2022)

Siskind, J.M. (1993), *Naive Physics, Event Perception, Lexical Semantics, and Language Acquisition,* In AI technical reports (1964-2004). Available at: https://dspace.mit.edu/handle/1721.1/6784 (Accessed: 22 Maj 2022)

Siskind, J.M. (2001), *Grounding the Lexical Semantics of Verbs in Visual Perception using Force Dynamics and Event Logic*, in Journal Of Artificial Intelligence Research, Volume 15, pp. 31-90, 2001. Available at: https://arxiv.org/abs/1106.0256 (Accessed 22 Maj 2022)

Subramanian, D. (2021), 'Speech to Text with Wav2Vec 2.0', *KdNuggets*, March. Available at: https://www.kdnuggets.com/2021/03/speech-text-wav2vec.html (Accessed: 5 September 2022)

Sugita, Y. and Tani, J. (2008), *A sub-symbolic process underlying the usage-based acquisition of a compositional representation: Results of robotic learning experiments of goal-directed actions*, in 2008 7th IEEE International Conference on Development and Learning, ICDL. Available at: https://ieeexplore.ieee.org/document/4640817 (Accessed: 16 September 2022)

Sutskever, I., Vinyals, O. and Le, Q.V. (2014), *Sequence to Sequence Learning with Neural Networks*, in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. Available at: https://arxiv.org/abs/1409.3215 (Accessed: 6 September 2022)

Tan, H. and Bansal, M. (2019), *LXMERT: Learning Cross-Modality Encoder Representations from Transformers*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, pp.5100–5111. Available at: https://aclanthology.org/D19-1514 (Accessed 1 July 2022)

Tiku, N. (2022), 'The Google engineer who thinks the company's AI has come to life', The Washington Post, 11 June 2022. Available at: https://www.washingtonpost.com/technology/2022/06/11/google-ai-lamda-blake-lemoine (Accessed: 1 July 2022)

Topan, S., Rolnick, D. and Si, X. (2021), *Techniques for Symbol Grounding with SATNet*, in 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Available at: https://proceedings.neurips.cc/paper/2021/hash/ad7ed5d47b9baceb12045a929e7e2f66-Abstract.html (Accessed: 12 June 2022)

Tuci, E., Ferrauto, T., Zeschel. A. and Massera, G., (2011) *An Experiment on Behaviour Generalisation and the Emergence of Linguistic Compositionality in Evolving Robots*, IEEE transactions on autonomous mental development, pp. 176–189. Available at: https://dx.doi.org/10.1109/TAMD.2011.2114659 (Accessed: 8 May 2022)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., and Polosukhin. I., Attention Is All You Need (2017), *Attention Is All You Need* in 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. Available at: https://arxiv.org/abs/1706.03762 (Accessed: 6 September 2022)

Wang P., Donti P.L., Wilder B. and Kolter Z. (2019) *Bridging deep learning and logical reasoning using a differentiable satisfiability solver*, Proceedings of the 36 th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Available at: https://arxiv.org/abs/1905.12149 (Accessed: 12 June 2022)

Wu, M., Wang, W. and Pan, S.J. (2020), *Deep Weighted MaxSAT for Aspect-based Opinion Extraction*, in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 5618–5628, November 16–20, 2020. Available at: https://aclanthology.org/2020.emnlp-main.453.pdf (Accessed: 6 September)

APPENDIX A - Dataset

LICENSE AND AVAILABILITY

This dataset is available at the address https://drive.google.com/drive/folders/1LptrlmdnqpO-LINNsJGG059HLnKToNvZ?usp=sharing and it is released under the license **Creative Commons Attribution-ShareAlike 3.0 Unported** (CC BY-SA 3.0) (creativecommons.org/licenses/by-sa/3.0).

THE LIST OF RECORDINGS

Below is the complete list of the 50 utterances recorded. Twenty voices (10 real and 10 artificial) were involved, therefore the dataset contained 20 x 50 = 1,000 recordings. For each audio recording a video was shot and then the two were paired and saved in a single MP4 file.

1	This is a pen
2	This is a pen
3	This is a pen
4	This is a pen
5	This is a pen
6	This is a phone
7	This is a phone
8	This is a phone
9	This is a phone
10	This is a phone
11	This is a spoon
12	This is a spoon
13	This is a spoon
14	This is a spoon
15	This is a spoon
16	This is a knife
17	This is a knife
18	This is a knife
19	This is a knife
20	This is a knife
21	This is a fork
22	This is a fork
23	This is a fork

24 This is a fork
25 This is a fork
26 Move the pen to the left
27 Move the pen to the right
28 Move the pen up
29 Move the pen down
30 Rotate the pen
31 Move the phone to the left
32 Move the phone to the right
33 Move the phone up
34 Move the phone down
35 Rotate the phone
36 Move the spoon to the left
37 Move the spoon to the right
38 Move the spoon up
39 Move the spoon down
40 Rotate the spoon
41 Move the knife to the left
42 Move the knife to the right
43 Move the knife up
44 Move the knife down
45 Rotate the knife
46 Move the fork to the left
47 Move the fork to the right
48 Move the fork up
49 Move the fork down
50 Rotate the fork

DATA AUGMENTATION

The following actions were applied to the original 1,000 audio/video samples of the dataset:

Audio

1. Decrease speed: multiply by a factor 0.7 (through Ffmeg utility)
2. Increase speed: multiply by a factor 1.3 (through Ffmeg utility)
3. Increase pitch: multiply by a factor 2 (through Rubberband utility)
4. Increase pitch: multiply by a factor 10 (through Rubberband utility)

5. Increase volume: multiply by a factor 2 (through Ffmeg utility)

Video (all through Ffmeg utility)

6. Flip:
    1. horizontally 1-25 and 28, 29, 30, 33, 34, 35, 38, 39, 40, 43, 44, 45, 48, 49, 50 (move up, move down and rotate)
    2. vertically 26, 27, 31, 32, 36, 37, 41, 42, 46, 47 (move to the right and to the left)
7. Increase brightness: add a factor 0.2
8. Decrease brightness: subtract a factor 0.1
9. Increase saturation: multiply by a factor 2
10. Zoom: zoom in (and crop accordingly) by a factor 1.2

This is the complete scheme of the data augmentation:

	Utterance	**Audio**					**Video**				
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
		Speed -	Speed +	Pitch +	Pitch -	Vol +	Flip H/V	Light +	Light -	Satur. +	Zoom +
1	This is a pen	-	+	+	-	+	H	+	-	+	+
2	This is a pen	-	+	+	-	+	H	+	-	+	+
3	This is a pen	-	+	+	-	+	H	+	-	+	+
4	This is a pen	-	+	+	-	+	H	+	-	+	+
5	This is a pen	-	+	+	-	+	H	+	-	+	+
6	This is a phone	-	+	+	-	+	H	+	-	+	+
7	This is a phone	-	+	+	-	+	H	+	-	+	+
8	This is a phone	-	+	+	-	+	H	+	-	+	+
9	This is a phone	-	+	+	-	+	H	+	-	+	+
10	This is a phone	-	+	+	-	+	H	+	-	+	+
11	This is a spoon	-	+	+	-	+	H	+	-	+	+
12	This is a spoon	-	+	+	-	+	H	+	-	+	+
13	This is a spoon	-	+	+	-	+	H	+	-	+	+
14	This is a spoon	-	+	+	-	+	H	+	-	+	+
15	This is a spoon	-	+	+	-	+	H	+	-	+	+
16	This is a knife	-	+	+	-	+	H	+	-	+	+
17	This is a knife	-	+	+	-	+	H	+	-	+	+
18	This is a knife	-	+	+	-	+	H	+	-	+	+
19	This is a knife	-	+	+	-	+	H	+	-	+	+

20	This is a knife	-	+	+	-	+	H	+	-	+	+
21	This is a fork	-	+	+	-	+	H	+	-	+	+
22	This is a fork	-	+	+	-	+	H	+	-	+	+
23	This is a fork	-	+	+	-	+	H	+	-	+	+
24	This is a fork	-	+	+	-	+	H	+	-	+	+
25	This is a fork	-	+	+	-	+	H	+	-	+	+
26	Move the pen to the left	-	+	+	-	+	V	+	-	+	+
27	Move the pen to the right	-	+	+	-	+	V	+	-	+	+
28	Move the pen up	-	+	+	-	+	H	+	-	+	+
29	Move the pen down	-	+	+	-	+	H	+	-	+	+
30	Rotate the pen	-	+	+	-	+	H	+	-	+	+
31	Move the phone to the left	-	+	+	-	+	V	+	-	+	+
32	Move the phone to the right	-	+	+	-	+	V	+	-	+	+
33	Move the phone up	-	+	+	-	+	H	+	-	+	+
34	Move the phone down	-	+	+	-	+	H	+	-	+	+
35	Rotate the phone	-	+	+	-	+	H	+	-	+	+
36	Move the spoon to the left	-	+	+	-	+	V	+	-	+	+
37	Move the spoon to the right	-	+	+	-	+	V	+	-	+	+
38	Move the spoon up	-	+	+	-	+	H	+	-	+	+
39	Move the spoon down	-	+	+	-	+	H	+	-	+	+
40	Rotate the spoon	-	+	+	-	+	H	+	-	+	+
41	Move the knife to the left	-	+	+	-	+	V	+	-	+	+
42	Move the knife to the right	-	+	+	-	+	V	+	-	+	+
43	Move the knife up	-	+	+	-	+	H	+	-	+	+
44	Move the knife down	-	+	+	-	+	H	+	-	+	+
45	Rotate the knife	-	+	+	-	+	H	+	-	+	+
46	Move the fork to the left	-	+	+	-	+	V	+	-	+	+
47	Move the fork to the right	-	+	+	-	+	V	+	-	+	+
48	Move the fork up	-	+	+	-	+	H	+	-	+	+
49	Move the fork down	-	+	+	-	+	H	+	-	+	+
50	Rotate the fork	-	+	+	-	+	H	+	-	+	+

APPENDIX B – Audio features extractor

This code is a modified version of the implementation by Dhilip Subramanian (2021) – available at www.kdnuggets.com/2021/03/speech-text-wav2vec.html – of the model Wav2vec developed by Baevski *et al.* (2020). It is available at the address github.com/fabiodeponte/symbol_grounding.

```
# Installing Transformer
!pip install -q transformers
!pip install -u librosa

# Library to to manage audio files
import librosa

#Importing Pytorch
import torch
import IPython.display as display

#Importing Wav2Vec
from transformers import Wav2Vec2ForCTC, Wav2Vec2Tokenizer

# Importing Wav2Vec pretrained model
tokenizer = Wav2Vec2Tokenizer.from_pretrained("facebook/wav2vec2-base-960h")
model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-base-960h")

# Reading taken audio clip
#display.Audio("taken_clip.wav", autoplay=True)
display.Audio("1artificial.wav", autoplay=True)

# Loading the audio file
audio, rate = librosa.load("voice.wav", sr = 16000) #16KHZ

# tokenizing
input_values = tokenizer(audio, return_tensors = "pt").input_values

# Storing logits (non-normalized prediction values)
logits = model(input_values).logits

# CALCULATING THE FEATURES VECTOR:
prediction = torch.argmax(logits, dim = -1)

# CONVERTING THE FEATURES VECTOR INTO A TEXT
transcription = tokenizer.batch_decode(prediction)[0]
```

APPENDIX C – Video features extractor

This code is a modified version of the implementation by Vladimir Iashin (2021) on his website iashin.ai/video_features/models/clip of CLIP model developed by Radford *et al.* (2021) at OpenAI (openai.com/blog/clip). It is available at the address github.com/fabiodeponte/symbol_grounding.

```
import pandas as pd
from numpy import save
from numpy import load
from models.clip.extract_clip import ExtractCLIP
from models.i3d.extract_i3d import ExtractI3D
from utils.utils import build_cfg_path, action_on_extraction
from omegaconf import OmegaConf
import torch

device = 'cuda' if torch.cuda.is_available() else 'cpu'

# Read the list of video filenames from "video.csv"
df = pd.read_csv("video.csv")

# Add a root directory to load the videos
root_dir = 'dir_containing_video_files/'
df.filename = root_dir + df.filename
videos = list(df.filename)

# Select the feature type
feature_type = 'clip'
model_name = 'ViT-B/32'

# Load and patch the config
args = OmegaConf.load(build_cfg_path(feature_type))
args.feature_type = feature_type
args.video_paths = videos.copy()
# args.show_pred = True
# args.stack_size = 24
# args.step_size = 24
# args.extraction_fps = 25
args.flow_type = 'raft'
# args.streams = 'flow'

# Load the model
extractor = ExtractCLIP(args)
modules_dict = extractor.load_model(device)
print(modules_dict.keys())

# Extract features
list_video_features=[]
for video_path in args.video_paths:
    print(f'Extracting for {video_path}')
    features = extractor.extract(device, modules_dict, video_path)
    a = list(features['clip'][0])
    list_video_features.append(a)

# Convert to DataFrame
list_video_features = pd.DataFrame(list_video_features)

# Save the features
save('video_features.npy', list_video_features)
```

APPENDIX D – Sequence-to-sequence model

This is a modified version of the code developed by Chollet (2017) available at https://keras.io/examples/nlp/lstm_seq2seq.
It is available at the address github.com/fabiodeponte/symbol_grounding.

TRAIN AND TEST

```
import numpy as np
import pandas as pd
import os
from numpy import save
from numpy import load
import matplotlib.pyplot as plt
from numpy import argmax
from tensorflow import keras
from keras import models
from keras.models import Sequential
from keras.layers import LSTM, TimeDistributed, RepeatVector, Dense
from keras.layers import Bidirectional
from keras.layers import Concatenate
from sklearn.model_selection import train_test_split
import tensorflow as tf
import tensorflow_addons as tfa
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score, precision_score, recall_score


# Load features
audio_features = pd.DataFrame(load('features/audio_wav2vec_features.npy',
allow_pickle=True))
video_features = pd.DataFrame(load('features/video_clip_features.npy',
allow_pickle=True))

# Convert audio features to one-hot
audio_features_onehot = keras.utils.to_categorical(audio_features,
num_classes=None, dtype='float32')

# Normalize video features, convert them to integers and then to one-hot
def normalize(x):
    return (x.astype(float) - video_features.min().min().astype(float)) /
(video_features.max().max().astype(float) -
video_features.min().min().astype(float))
def de_normalize(x):
    return (x * (video_features.max().max().astype(float) -
video_features.min().min().astype(float)) +
video_features.min().min().astype(float)).astype(int)
video_features_norm = normalize(video_features)
video_features_norm_100 = video_features_norm * 100
video_features_norm_100 = video_features_norm_100.round()
video_features_onehot = keras.utils.to_categorical(video_features_norm_100,
num_classes=None, dtype='float32')

# Split train and test
X_train, X_test, y_train, y_test = train_test_split(video_features_onehot,
audio_features_onehot, test_size=0.2, random_state=True)
```

```
# Create decoder_input_data, decoder_target_data, encoder_input_data
encoder_input_data = X_train
decoder_input_data = y_train
decoder_target_data = np.zeros(decoder_input_data.shape)
for a in range(decoder_input_data.shape[0]):
    for i in range(decoder_input_data.shape[1]-1):
        decoder_target_data[a][i]=decoder_input_data[a][i+1]
    decoder_target_data[a][decoder_input_data.shape[1]-1] =
decoder_input_data[a][decoder_input_data.shape[1]-1]

encoder_input_data_test = X_test
decoder_input_data_test = y_test
decoder_target_data_test = np.zeros(y_test.shape)
for a in range(y_test.shape[0]):
    for i in range(y_test.shape[1]-1):
        decoder_target_data_test[a][i]=y_test[a][i+1]
    decoder_target_data_test[a][y_test.shape[1]-1] = y_test[a][y_test.shape[1]-
1]

# Save lengths of vectors
num_encoder_tokens = encoder_input_data.shape[2]
num_decoder_tokens = decoder_input_data.shape[2]
target_max_length = decoder_input_data.shape[1]


# === SET UP THE MODEL ===

batch_size = 64  # Batch size for training.
epochs = 100  # Number of epochs to train for.
latent_dim = 1024  # Latent dimensionality of the encoding space.


# Define an input sequence and process it.
encoder_inputs = keras.Input(shape=(None, num_encoder_tokens))
encoder = keras.layers.LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)

# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]

# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = keras.Input(shape=(None, num_decoder_tokens))

# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = keras.layers.LSTM(latent_dim, return_sequences=True,
return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs,
initial_state=encoder_states)
decoder_dense = keras.layers.Dense(num_decoder_tokens, activation="softmax")
decoder_outputs = decoder_dense(decoder_outputs)

# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = keras.Model([encoder_inputs, decoder_inputs], decoder_outputs)

# Compile
```

```
model.compile(optimizer="adam", loss="categorical_crossentropy",
metrics=["accuracy", keras.metrics.Precision(thresholds=None, top_k=None,
class_id=None, name=None, dtype=None), keras.metrics.Recall()])




# === TRAIN THE MODEL ===
results = model.fit(
    [encoder_input_data, decoder_input_data],
    decoder_target_data,
    batch_size=batch_size,
    epochs=epochs,
    validation_split=0.2,
)

# Plot the loss curves graph
history = pd.DataFrame(results.history)
fig = plt.figure
plt.plot(history['loss'], label="training loss");
plt.plot(history['val_loss'], label="validation loss");
plt.legend();
plt.show()

# Plot the accuracy/precision/recall curves graph
history = pd.DataFrame(results.history)
fig = plt.figure
plt.plot(history['accuracy'], label="training accuracy");
plt.plot(history['val_accuracy'], label="validation accuracy");
plt.plot(history['precision'], label="training prediction");
plt.plot(history['val_precision'], label="validation prediction");
plt.plot(history['recall'], label="training recall");
plt.plot(history['val_recall'], label="validation recall");
plt.legend();
plt.show()

# Save model
model.save("MODEL_weights")


# === TEST THE MODEL ===

# Evaluate the model against the test dataset
model_eval = model.evaluate([encoder_input_data_test,
decoder_input_data_test],decoder_target_data_test)

# Print the results
print("TEST MODEL\nLoss:", model_eval[0], "\nAccuracy:", model_eval[1], "\
nPrecision:", model_eval[2], "\nRecall:", model_eval[3])
```

PREDICT SEQUENCES FROM X_TEST AND X_TRAIN

```
# ==== PREDICT SEQUENCES ====
model = keras.models.load_model("MODEL_weights")

# define euclidean distance
from math import sqrt

# calculate euclidean distance
def euclidean_distance(a, b):
      return sqrt(sum((e1-e2)**2 for e1, e2 in zip(a,b)))

# define model
encoder_inputs = model.input[0]  # input_1
encoder_outputs, state_h_enc, state_c_enc = model.layers[2].output  # lstm_1
encoder_states = [state_h_enc, state_c_enc]
encoder_model = keras.Model(encoder_inputs, encoder_states)

decoder_inputs = model.input[1]  # input_2
decoder_state_input_h = keras.Input(shape=(latent_dim,))
decoder_state_input_c = keras.Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_lstm = model.layers[3]
decoder_outputs, state_h_dec, state_c_dec = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs
)
decoder_states = [state_h_dec, state_c_dec]
decoder_dense = model.layers[4]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = keras.Model(
    [decoder_inputs] + decoder_states_inputs, [decoder_outputs] + decoder_states
)


# declare the function that extracts the sequence
def decode_sequence(input_seq):
    states_value = encoder_model.predict(input_seq)
    target_seq = np.zeros((1, 1, num_decoder_tokens))
    stop_condition = False
    decoded_sentence = ""
    predicted_tokens=[]
    i=0
    while not stop_condition:
        i=i+1
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value,
verbose=False)
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        predicted_tokens.append(sampled_token_index)
        if i == target_max_length:
            stop_condition = True
        target_seq = np.zeros((1, 1, num_decoder_tokens))
        target_seq[0, 0, sampled_token_index] = 1.0
        states_value = [h, c]
    return predicted_tokens



# PREDICT y_TRAIN AND CALCULATE COSINE DISTANCE
entire_test_sequence_length=len(encoder_input_data_test)
predicted = []
target=[]
```

```
sum_distance = 0
for seq_index in range(entire_test_sequence_length):
    input_seq = encoder_input_data[seq_index : seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)
    print("-SAMPLE N.", seq_index)
    print("Predicted audio features:", decoded_sentence)
    print("---Target audio features:",
list(argmax(decoder_target_data[seq_index], axis=1, out=None)))
    predicted.append(decoded_sentence)
    target.append(list(argmax(decoder_target_data[seq_index], axis=1,
out=None)))
    distance = euclidean_distance(decoded_sentence,
list(argmax(decoder_target_data[seq_index], axis=1, out=None)))
    print("Euclidean distance:", distance)
    sum_distance = sum_distance + distance
mean_distance = sum_distance / entire_test_sequence_length
print ("\n======\nMEAN DISTANCE:", mean_distance)

# Save the predictions
save('PREDICTED_MODEL_TRAIN_Ypred.npy', pd.DataFrame(predicted))
save('PREDICTED_MODEL_TRAIN_Ytarget.npy', pd.DataFrame(predicted))


# PREDICT y_TEST AND CALCULATE COSINE DISTANCE
entire_test_sequence_length=len(encoder_input_data_test)
predicted = []
target=[]
sum_distance = 0
for seq_index in range(entire_test_sequence_length):
    input_seq = encoder_input_data_test[seq_index : seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)
    print("-SAMPLE N.", seq_index)
    print("Predicted audio features:", decoded_sentence)
    print("---Target audio features:",
list(argmax(decoder_target_data_test[seq_index], axis=1, out=None)))
    predicted.append(decoded_sentence)
    target.append(list(argmax(decoder_target_data_test[seq_index], axis=1,
out=None)))
    distance = euclidean_distance(decoded_sentence,
list(argmax(decoder_target_data_test[seq_index], axis=1, out=None)))
    print("Euclidean distance:", distance)
    sum_distance = sum_distance + distance
mean_distance = sum_distance / entire_test_sequence_length
print ("\n======\nMEAN DISTANCE:", mean_distance)

# Save the predictions
save('PREDICTED_MODEL_TEST_Ypred.npy', pd.DataFrame(predicted))
save('PREDICTED_MODEL_TEST_Ytarget.npy', pd.DataFrame(predicted))
```

CONVERT PREDICTED SEQUENCES TO TEXT

```
import pandas as pd
from numpy import load
import torch
from transformers import Wav2Vec2ForCTC, Wav2Vec2Tokenizer
import IPython.display as display

# Importing Wav2Vec pre-trained model
tokenizer = Wav2Vec2Tokenizer.from_pretrained("facebook/wav2vec2-base-960h")
model = Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-base-
960h")ned("facebook/wav2vec2-base-960h")



original_features = pd.DataFrame(load('PREDICTED_MODEL_TRAIN_Ytarget.npy',
allow_pickle=True))
predicted_features = pd.DataFrame(load('PREDICTED_MODEL_TRAIN_Ypred.npy',
allow_pickle=True))

out = []
for i in range(len(original_features)):
    new_row=[tokenizer.batch_decode(torch.tensor([original_features.iloc[i]]))
[0]]
    out.append(new_row)

pd.set_option('display.max_rows', None)
print(pd.DataFrame(out))
```