École Normale Supérieure

Département d'Informatique

Rapport de Stage de L3

Multimodal Learning

Examples in Gesture and Audio-Visual Speech Recognition

Hsieh Yu-Guan

Sous la direction de

Mathieu Lefort[1] & Amélie Cordier[2]

[1]LIRIS, Équipe SMA

[2]Hoomano, Équipe R&D

14 Juin 2017 – 11 Août 2017

1 Introduction

1.1 Environment, material and software framework

Mon stage s'inscrit dans le cadre du laboratoire commun BEHAVIORS.AI qui est un projet de recherche collaboré entre la société Hoomano et LIRIS. Hoomano est une entreprise créée en 2014 dédiée au développement de logiciel de robots sociaux (ex: Nao et Pepper) et son moteur d'interaction vise à offrer une interaction homme-robot la plus naturelle possible. Le LIRIS (Laboratoire en InfoRmatique, Image et Système d'Information) est une unité mixte de recherche en informatique située à Lyon qui compte 14 équipes de recherche et 330 membres.

Pendant mon stage, j'ai passé les deux premières semaines au laboratoire et pour le reste du temps j'étais à Hoomano car j'entrînais les modèles sur le serveur de l'entreprise. Tous les codes ont été réalisés en python et précisément à l'aide de la biblithèque open source TensorFlow développé par Google. Cela incluait la construction des différents architectures et réseaux de neurones, l'entraînement et l'évaluation. TensorBoard était un outil assez pratique pour visualiser l'apprentissage, et je pouvais même facilement visualiser les données et les représentation apprises grâce au plugin qui est offert. Toute la suite du rapport sera rédiger en anglais dû à la difficulté de traduire tous les terms techniques en français.

1.2 Scientific overview

Historically, expert systems have been widely used for the design of artificial intelligence. Agents rely heavily on handcrafted ontologies and symbolic rules to make decisions and act in the world. Although impressive results have been obtained with this line of research, it requires a lot of engineering efforts and human knowledge. Furthermore, engineering in this way all the behaviors that are needed to display general human level intelligence is out of reach.

On the other hand, we face also the symbol grounding problem [11]. The symbols manipulated by the AI systems have no meaning to them. To solve these problems, the embodied paradigm [6] first argues an agent must be able to perceive and act from its own perspective (for example, it should be a robot). Developmental robotics [50] further emphasizes the importance of endowing the robot with the capacity of learning new knowledge from its own sensorimotor experience.

In [40], Smith and Gasser discerned six fundamental principles for the development of embodied intelligence: mulimodality, incremented development, physical interaction with the environment, exploration, social guidance and symbolic language aquisition. I was particularly insterested in the multimodal point in my internship. The everyday concept that a human acquires is intrinsically multimodal. For example, the word "cat" can be quickly associated with the appearance of a cat, its vocalization, and the tactile feeling that we have while petting a cat. The existence of neurons that receive early information from different modalities have also been proven [28].

On the other hand, recently, deep learning has achieved a great success in various domains, such as image recognition [18], text generation [10] or machine translation [45]. It has also been more and more often applied to mutlimodal input [2, 33]. Its capacity of learning a hierarchical representation of data in a fully unsupervised way [39, 49] is particularly interesting in the domain of robotics.

In my internship, I studied mainly the application of multimodal learning using deep networks in the fields of multimodal gesture recognition and audio-visual speech recognition (AVSR). I focused especially on the problem of shared representation learning and knowledge transfer. I wanted to show that the availablity of multiple modalities could enable the model to learn a better representation for each single modality and that one can leverage information from one modality to be used for another when there is an imbalance of amount of data among different sources.

The report is organized as follow. In Section 2 I briefly review related work on multimodal machine learning, gesture recognition and AVSR. In section 3, I present several basic deep neural network architectures that played an important role in my work. In section 4 and 5, I describe respectivly the datasets and common experimental setups that were used in my internship. Experimental details and results are given in section 6 and 7. Finally, conclusions and perspectives are presented in section 8.

The source code of all of the work described here can be found on my github: https://github.com/cyber-meow/internship_2017.

2 Related Work

2.1 Multimodal learning

[2] offers an overview of recent advances in multimodal machine learning. In [33], the authors use a deep network to learn a joint representation of visual and auditory input. They show that better features for one modality can be learned if multiple modalities are present at feature learning time. There has also been a surge of interest in the exploitation of multimodal information in the fields of image annotation [51], zero-shot learning [9, 41], and automatic image caption generation [15]. They first train visual and language models separately and as a next step they try to learn either a common embedding [51], a mapping [9, 41] or an alignment [15] between the two models.

Applications in the field of robotics can also been found. [8] proposes a network that is able to learn both a symbolic representation of data and a fine discrimination between two similar stimuli in an unsupervised way. The authors apply their method to visual, audiotory and proprioceptive data of the humanoid robot iCub. A multimodal embedding that is able to combine information coming from vision, language and motion trajectories is defined in [44] and endows the robot with the capacity of manipulating an unseen object.

2.2 Gesture recognition

Gesture recognition has been studied for a while within the fileds of computer visoin and pattern recognition [26, 43]. Recently, deep learning algorithms have led to significant advances in this domain [1]. The use of Convolutional Neural Networks (CNNs) is the most common [30]. 3d CNNs are used to deal with image sequences in [27]. Architectures that take in multimodal inputs have also grown in popularity. [31] copes with color, depth, skeleton and audio information using CNNs and Recurrent Neural Networks (RNNs). Others use 3d CNNs and deep belief networks (DBNs) while facing similar multimodal inputs [32, 36, 52].

2.3 Audio-visual speech recognition

AVSR is probably one of the earliest examples of multimodal research. Most early works were based on various hidden Markov model (HMM) extensions [37], but the use of neural networks were also explored [5, 53]. Although research into AVSR is not as common these days, it has drawn attention from the deep learning community [16, 33, 34]. A transfer deep learning framework applied in AVSR proposed in [29] forms the base of my study in 7.2. [7] and [12] give several examples of how deep architectures can be used to deal with audio data.

3 Presentation of Basic Network Architectures

3.1 Convolutional Neural Network

CNNs are an early family of deep learning architectures inspired from the human vision system [19]. Generally we have convolutional layers alternating with pooling (subsamping) layers, but fully connected layers can also be introduced (see Figure 5). CNNs have been shown to achieve state-of-the-art performance in image processing tasks such as image classification [18] and object detection [20]. However they can be equally applied in other fields like speech recogntion [7].

For a convolutional layer with a mono-channel input x, the latent representation of the k-th feature map is given by

$$h^k = \sigma(x * W^k + b^k)$$

where W^k is the k-th kernel, b^k the bias is broadcasted to the whole feature map, $*$ denotes the 2d convolution and σ is an ativation function. Currently, ReLu (Rectified Linear Units) is probably the one that is the most commonly used for a CNN [18]. It is defined by

$$\sigma(x) = x^+ = max(0, x),$$

when x is a single scalar and otherwise the above function is applied to each element of the input. The convolution kernel is extended to the full depth of input when there are multiple input channels. On the contrary, no parameters are required for defining a pooling layer. It just take some $d \times d$ region (supposing that the kernel has the same height and width) and output a single value, which for example, is the maximum in that region when using max-pooling.

There are yet two other important factors that are used to define these operations: *stride* and *padding*.[1] Concerning this, two different zero paddings are defined in TensorFlow, 'SAME' and 'VALID'. For 'SAME' padding, $h_o = \lceil h_i/s_h \rceil$, and for 'VALID' padding, $h_o = \lceil (h_i - h_f + 1)/s_h \rceil$, where h_i, h_o, h_f are respectivly the height of the input feature map, the output feature map and the filter (kernel) and s_h is the stride of the height dimension. Similar formulas also exist for the width.[2]

Suppose all the parameters of a CNN model have been learned, the model is then ready to be used for a specific task. One just needs to run the model forward as decribed above. Therefore, the purpose of the learning phase is to find the appropriate parameters that can help us solve the task. Mathematically, this is achieved by optimizing the weights of the network to minimize some loss function (e.g. cross entropy or L2 distance, see section 5). The optimization is usually done by a SGD-based algorithm. SGD is the abbreviation of stochastic gradient descent. At each training step, we compute the gradients of the loss with respect to all weights in the network and then update the weights according to these gradients. Using the chain rule, gradients are computed layer by layer (starting from the output of the network) and the error is thus *back-propagated* to the whole network.[3]

In [14], 3d CNNs are proposed to be used for video inputs. They're like traditional CNNs except for the fact that we replace 2d feature maps by 3d feature maps and 2d kernels by 3d kernels.

[1] Due to the space limit, more details of the CNN architecture, including the role of these two hyperparameters, can be found here: http://cs231n.github.io/convolutional-networks/.

[2] https://www.tensorflow.org/api_guides/python/nn#Convolution.

[3] http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/.

3.2 Auto-Encoder

Auto-Encoders are networks that are trained to minimize the reconstruction error by back-propagating it from the output layer to hidden layers. In the simplest model with one hidden layer, an auto-encoder takes an input $\mathbf{x} \in \mathbb{R}^d$ and maps it to the latent representation $\mathbf{h} \in \mathbb{R}^{d'}$ given by $\mathbf{h} = \sigma(W\mathbf{x} + \mathbf{b})$ where W is a weight matrix, $\mathbf{b}$ is a bias vector and σ is an activation function. Then the network tries to reconstruct the input by a reverse mapping $\mathbf{x}' = \sigma(W'\mathbf{h} + \mathbf{b}')$. The loss function can be for instance $||x' - x||$ where $||.||$ is some distance.

To prevent the auto-encoder from learning the identity function as a trivial solution, several regularization techniques have been proposed. The bottleneck approach forces dimensionality reduction by having fewer neurons in hidden layers than in the input layer. For example, in the above case, we must have $d' < d$. Sparse auto-encoders impose sparsity on hidden units [21]. Denoising auto-encoders, which played an important role in my work, try to recontruct the clean input from its corrupted version [3, 48]. Binomial noise (switching pixels on or off) adding to input or hidden layers were used in my case.

Intuitively, auto-encoders are useful for data reconstruction. Nevertheless, the true interest lies in fact in its capacity to learn a representation (encoding) for a set of data in a purely unsupervised fashion [49]. Recently, auto-encoders have also been more and more used as a generative model [4].

3.3 Convolutional Auto-Encoder

Fully connected auto-encoders ignore the 2d image structure. This can cause problems when dealing with real-world size inputs, and introduce redundancy in the parameters. Convolutional Auto-Encoders [22, 47] (CAEs) are intuitively similar to architectures described in 3.2. However, convolutional and transposed convolutional layers are used instead. Transposed convolutional layers are also called deconvolutional layers [54]. We perform in fact also a convolution operation but with zero paddings around the image and sometimes around each pixel to upsample the image and to inverse the previous convolution (imagine that if the value of one pixel comes from 9 pixels of the previous layer, when doing a transposed convolution this pixel contributes to the same 9 pixels for reconstruction). This github directory https://github.com/vdumoulin/conv_arithmetic is quite useful for the understanding of the concept.

Pooling and unpooling layers are also sometimes considered in a CAE architecture [47]. More details on this aspect, which was not used in my work, can be found in in [35, 47].

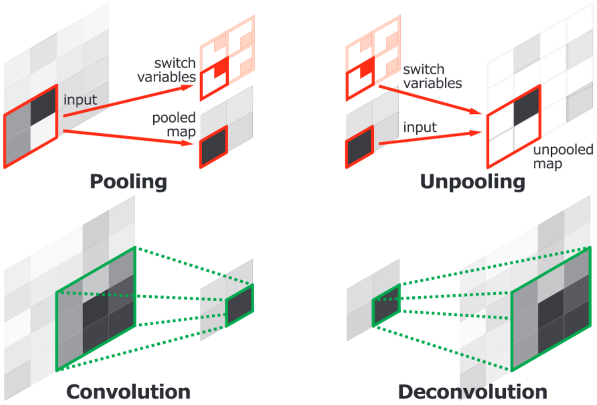

Figure 1: Illustration of deconvolution and unpooling layers [35]

4 Datasets and Preprocessing

Many datasets were explored during my internship. In this report, I will only detailed the three that I used the most. Two of them are for gesture recognition: Creative Senz3D [24, 25] and ASL Finger Spelling [38], and one is for AVSR: AVLetters [23].

4.1 Creative Senz3D

The dataset contains gestures coming from 4 different people, each performing 11 different static gestures repeated 30 times each, for a total of 1320 samples. For each sample, color, depth and confidence frames are available. I only used the color and depth frames of this dataset since my architectures take at most two modalities in input. The original size of each image is 480 × 640 and they're resized to 299 × 299 pixels before being fed to the network since this is also the input size of the Inception model [46]. No other preprocessing are done. For both color and depth images I use the three color channels (even though a priori only one channel is needed for depth maps).

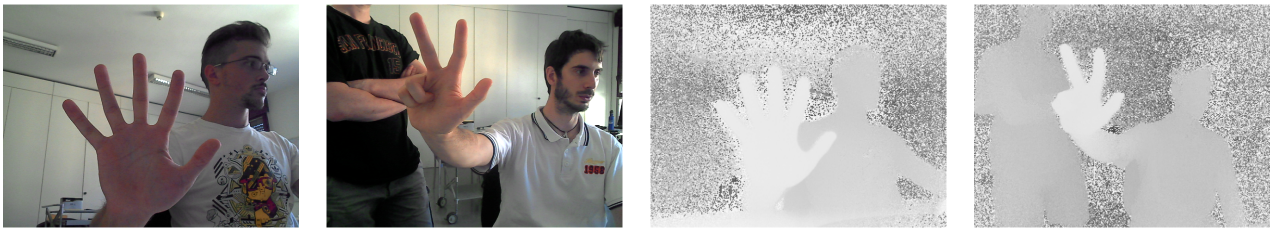

Figure 2: **Example images in the Creative Senz3D dataset.**
Left Two) Color images.
Right Two) Corresponding depth images.
All of the images are of size 480 × 640 and contain the the entire upper body of the subject.

4.2 ASL Finger Spelling

The dataset is composed of more than 60000 images in each modality (RGB and depth images are provided). Five subjects are asked to perform the 24 static signs in the American Sign Language (ASL) alphabet (excluding j and z which involve motion) a certain number of times, captured with similar lighting and background.

Images of this dataset are of variable sizes. The data preprocessing includes resizing each image to 83 × 83 pixels, converting them to grayscale and Z-normalization (normalizing to zero mean and unit of variance).

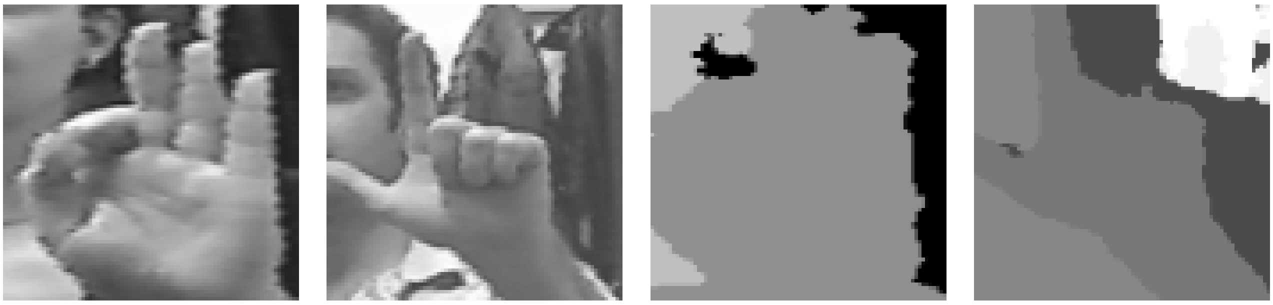

Figure 3: **Example images in the ASL Finger Spelling dataset (after preprocessing).**
Left Two) Grayscale intensity images.
Right Two) The corresponding depth maps.
Images of this dataset have variable sizes, and they're all resized to 83 × 83 before being fed to the network. Generally only the hand region is contained in image.

4.3 AVLetters

The dataset comprises video and audio recordings of 10 speakers uttering the letters A to Z, three times each. We count therefore 780 samples in total. For video data, image sequences of pre-extracted lip regions are provided. Each single image if of size 60 × 80. For audio data, only the mel-frequency cepstrum coefficients (MFCCs) are given, and each audio frame is represented by 26 MFCCs. The lack of raw audio data is a strong constraint on what we're able to do on this dataset.

Since all utterances don't have the same time duration, I used fourier resamping to force every video input to be of length 12 and every audio input to be of length 24. Video frames are Z-normalized. Several data augmentation techniques are also considered, including random brightness adjusting, random contrast adjusting and random cropping (but at least 80% of the original image is kept).

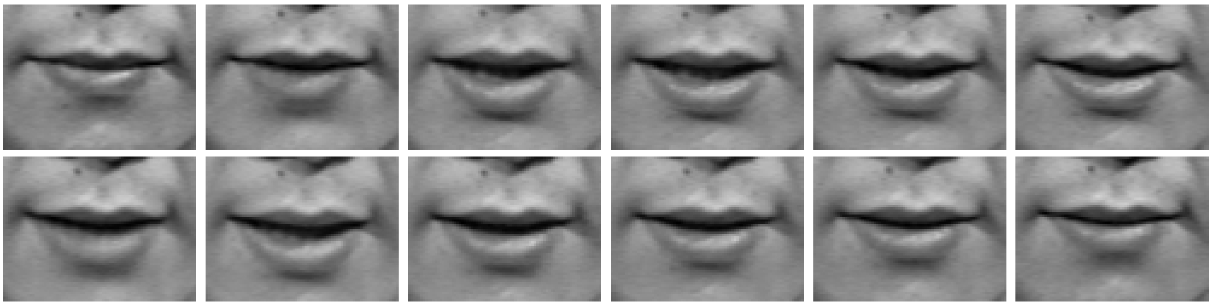

Figure 4: Example visual input for the AVletters dataset (left to right, top to bottom). Pre-extracted lip regions of 60 × 80 pixels are provided. Each image sequence is resampled to be of length twelve in order to give an input of fixed size to the network.

5 Experimental Setup

To train a classifier I employed the cross entropy cost function and to train an auto-encoder the L2 distance between the input and output vector was used as the loss. For the sake of preventing overfitting, L2 regularization [3] was applied to all the weights of network with a regularization coefficient η. The Adam algorithm [17] was then introduced for minimizing the loss function. An exponential decay was further used for the stepsize α of this algorithm with an initial stepsize α_0 varying from 0.01 to 0.0001 depending on experiment. The decaying rate γ was generally close to 0.8 and the decay takes place every 100 training steps.

Inputs of the network were normally fed as mini-batches of size 24 (smaller and bigger batch sizes were also experimented). Batch normalization [13] were introduced after every convolutional and transposed convoluitonal layer. Therefore, the real operations used to compute neural activations are more complicated then what are described above. The used settings and hyperparameter values aren't necessarily optimal since I wasn't able to test all the possible combinations.

Here are some more details of the network architectures: ReLu (Rectified Linear Unit) activations were added to all the hidden layers [18] while no activation function was used for the output layer. For classification model dropout [42] was always applied to the second to last layer during training. The output of the classification layer was mapped to the probabilities that one data example belongs to each class by the softmax function.

For classification experiments, except for the one described in 7.2, the dataset was always separated into training and test set. The classifier was first trained on the training data and then tested on test data once the training phrase was finished. Unless stated otherwise, the prediction accuracies mentioned hereinafter were always evaluated on the test set.

Moreover, we can consider two different possible settings:

Subject Dependent. In a subject dependant setting, data samples are separated randomly into training set and test set. Therefore, during the testing phase, the classifier does not need to deal with data from an individual that it has never seen before.

Subject Independant. On the contrary, the classifier faces individuals never seen before during testing in a subject independant setting. In my case, I always separated one subject from the others to form the test set while the training set consisted of the rest of the data.

6 Experiments and Results: Unimodal Cases

6.1 Classification

For every dataset, I began with training a classifier on it in a totoally supervised manner. This gave me an insight into its data quality, the preprocessing effectiveness and ensured that further experiments could be conducted. CNN is then one of the most suitable architecture for this purpose.

First, it worth mentioning the problem of overfitting. It was observed for almost all the classification experiments that I carried out, and it was particularly severe for CNNs with many layers. In fact, CNN architectures could usually classify perfectly the training data, but experienced a drop of performance when evaluating on test set.

It is well known that by reducing the number of hidden layers and increasing the weight regularization coefficient λ we may be able to cure this problem. For example, when λ was augmented from 0.0004 to 0.1, the classification accuracy for the audio input of the AVLetters dataset increased by about 10 points (from 63.22% to 77.84%) while using the architecture detailed on the left side of Figure 10. By using fewer hidden layers in the 3d CNN architecture overfitting was also alleviated when dealing with the video input of this same dataset. However, these techniques did not always work and more often I got a poorer performance during training without an improvement of performance for test.

Below I'll briefly describe the final results of this part dataset by dataset.

6.1.1 Creative Senz3d

The tested architectures were perceptron (in all of this report by perceptron I denote a single-layer perceptron), pre-trained InceptionV4 model [46] and several hand-coded CNNs[4]. The lack of data quantity, variety, and the fact that the head is also contained in the image seems to increase the classification difficulty. In conlusion, I found that it was more interesting to use a subject dependent setting for this dataset. In this case, for RGB images, all of the classifiers were able to have a classification accuracy that is closed to 100%. For depth images, the classification accuracy was between 60% and 70% using a perceptron and near 90% for other architectures.

6.1.2 ASL Finger Spelling

The large number of data contained in this dataset and the relatively simple image content (single hand instead of the entire upper body) makes the classification task much easier. By using the CNN architecture shown in Figure 5, I could achieve a classification accuracy of respectively 79.73% and 64.46% for intensity and depth images (see Table 1) in a subject independant setting (four subjects for training and one subject for testing). Fewer layers in the CNN architecture may also allow us to achieve the same performance. More experiments need to be carried out to find the most suitable hyperparameters.

[4] For those who are interested, please refer to my github directory for the used hyperparameters.

6.1.3 AVLetters

One can refer to Figure 10 for the main CNN architectures that were used in this dataset. Notice that 3d CNNs were employed to deal with video inputs. Considering the small number of available data, a speaker dependent setting was used. With some carefully chosen hyperparameter values and network architectures (also see Figure 10), the prediction accracy was of 77.84% for audio and 54.32% for video.

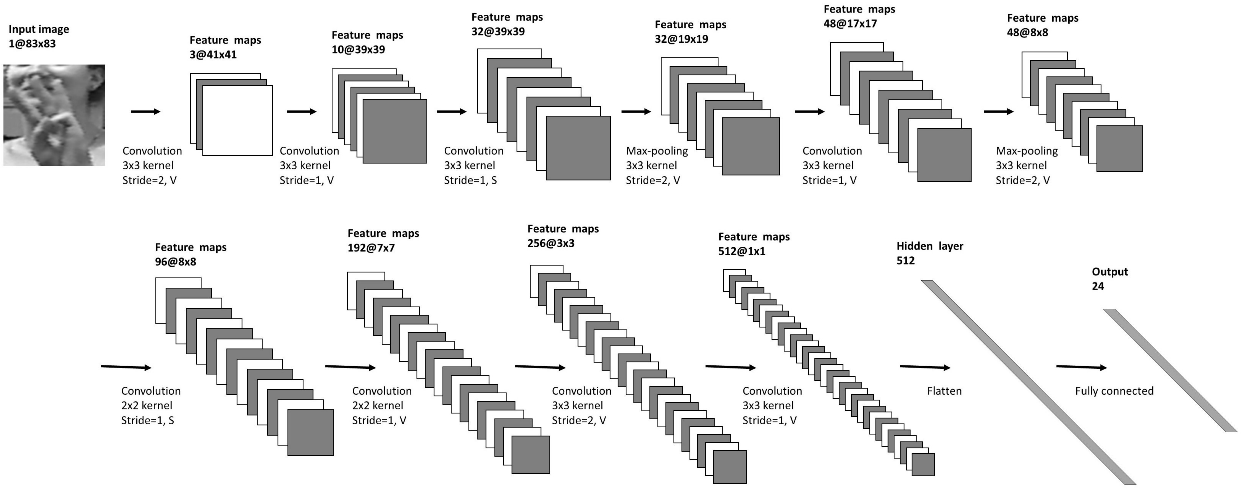

Figure 5: CNN architecture used for the Finger Spelling dataset.
The input of the nework is a one-channel image of size 83 × 83. It contains ten hidden layers. S stands for 'SAME' padding and V stands for 'VALID' padding (see text).

6.2 Convolutional auto-encoder

Several distinct CAE architectures were tested in my internship. Here I present the one with five hidden layers; therefore it contains three convolutional layers and three transposed convolutional layers as illustrated in Figure 6. The end-to-end training instead of a greedy layer-wise approach was employed.

The proposed architecture was then trained on the two gesture recognition datasets. First of all, I was interested in the denoising capcity of the auto-encoder. An example is given in Figure 7. The auto-encoder is effectively able to reconstruct the clean image in a way, even though the result is blurred and sometimes distorted.

What's more important, can meaningful high-level features be learned in this way? The output of the middle layer was taken as a new representation of the input data. By doing principal component analysis, it could be projected into three dimensions for visualization. However, to quantify the learned features, they're further used for classification by training a perceptron on top of it.

As suggested in 6.1, I used a subjetdependant setting for Creative Senz3d while a subject independant setting was employed for ASL Finger Spelling. I compared this new classifier with a perceptron built on raw data input. As already mentioned earlier, the latter could classify perfectly the input when dealing with color images of Creative Senz3d. In other cases, I observed always an improvement of 10-20% for the prediction accuracy thanks to the use of the learned representation of the data (refer to Table 1 for results on ASL Finger Spelling). Useful high-level data representation can thus be learned in a totally unsupervised manner.

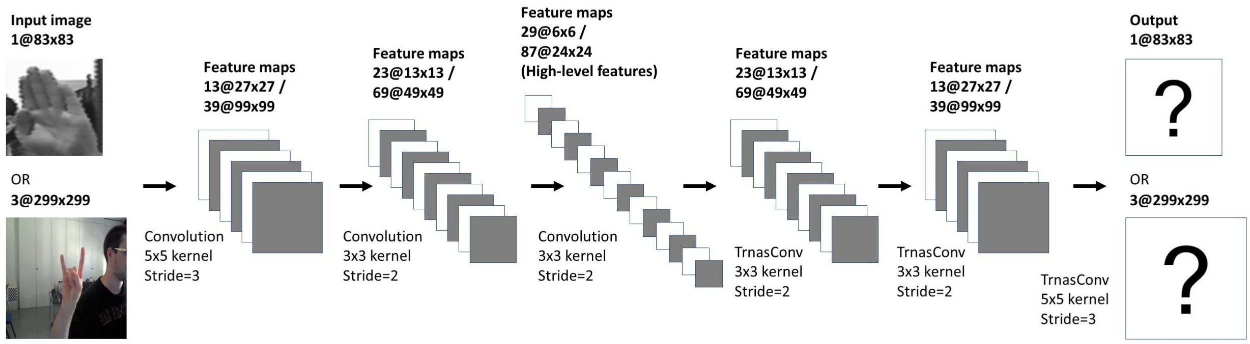

Figure 6: Convolutional auto-encoder architecture with three convolutional layers and three tranposed convolutional layers.
Activation values of the middle layer are taken as high-level features of the input image. Inputs of the network can be of different sizes. We only use 'VALID' paddings here.

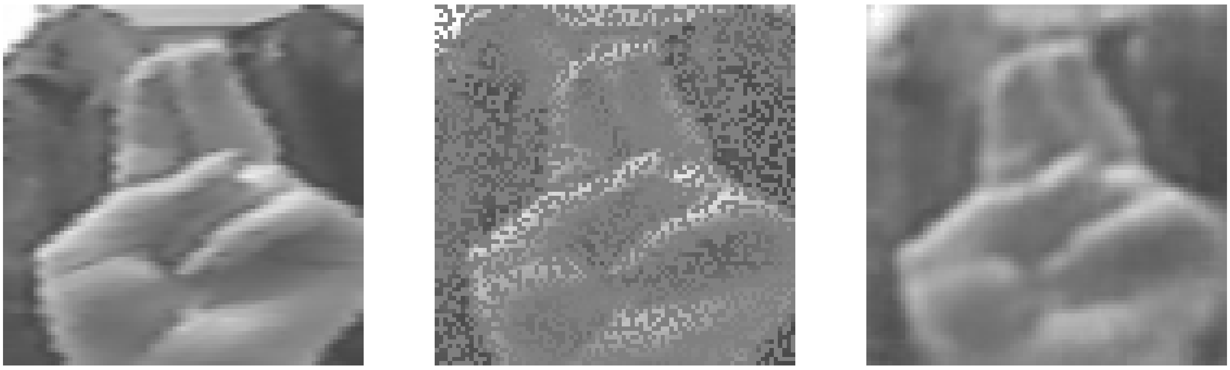

Figure 7: Image restoration using convolutional auto-encoder.
Left) Clean Image.
Middle) Noisy image [input].
Right) Restored image [output].

7 Experments and Results: Multimodal Cases

7.1 Shared representation learning

A first fundamental chanllenge in multimodal learning is the problem of representing and summarizing data from several modalities. How can we relate information from multiple sources? Mainly inspired from [8, 33], I generalized the CAE architecture introuduced in 6.2 to a multimodal setup. As presented in Figure 8, two CAEs of different modalities share a common middle layer by doing a sum of corresponding values. I first pre-train the first two layers of each modality using an unimodal CAE. In a second stage, I train the rest of the network to reconstruct the two modalities of the input.

To prevent the network from finding representations such that different hidden units are tuned for different modalities separately, random dropouts are added to inputs in such a way that only by combining the two modalities we have 100% of the input information. In particular, sometimes one modality can be totally absent whereas the whole clean image is given for the other modality. In addition, proper scaling were also considered to keep the expected sum of the activations at each layer to be the same.

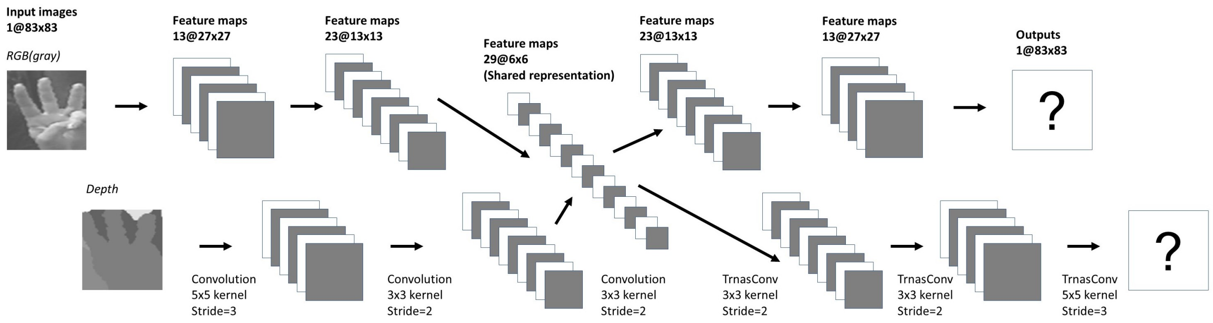

Figure 8: **The bimodal convolutional auto-encoder model that is used to learn shared multimodal representation.**
We simply take the CAE architecture that is introduced Figure 6 for each modaliy but force them to have a shared middle layer by adding the corresponding activation values. We then try to reconstruct the two images separately through two disjoint paths.

Ideally, we expect that the network is able to capture correlations across different modalities and can thus also learn a better single modality representation. To verify this hypothesis, I built a perceptron on the top of the middle layer of the architecture and trained it in a supervised way while only one modality was given in input. For example, if I wanted to train a classifier for color images, zeros were fed as depth inputs. The results are shown in Table 1. Unfortunately, the presence of the two modalities during feature learning doesn't bring a significant improvement and seems to be useless.

How about exploiting the information from the two modalities in a totally supervised way? I took the CNN architecture of Figure 5 for color and depth images separately until the seventh layer where a fusion was carried out. After training the network, the prediction accuracy was always at about 80% and no improvement was obtained compared with a CNN trained only on color inputs. To conclude, color and depth images are probably two modalities that are too similar. Color images contain already all the necessary information for the task at hand so that depth maps don't bring any supplementary information that benefits the specifique purpose I defined.

However, multimodal information may still be useful when one or several modalities are noisy whereas clean information are available for the others. For example, I trained a network to construct clean color images from noisy depth maps. When I did the classification based on the learned representation, the performance was as if I trained directly a depth CAE.

Table 1: **Classification performance on the ASL Finger Spelling dataset**
Raw) Perceptron that reads raw input data.
CAE features) Perceptron stacked on the middle layer of the CAE. (6.2).
Shared) Perceptron that exploits the shared representation learned by a bimodal CAE (7.1).
CAE architecture) Perceptron stacked on the middle layer of the CAE but train the whole network in a supervised way as a CNN.
CNN) The CNN architecture in Figure 5 (6.1.2).
The used hyperparameters are $\alpha_0 = 0.005$, $\gamma = 0.8$ and $\eta = 0.0004$. We notice that we have exactly the same network architecture for the middle three exerimental setups and only the training process differs one from another.

		Raw	CAE features	Shared	CAE architecture	CNN
Intensity	train	69.47 %	78.87 %	85.85 %	91.29 %	99.69 %
	test	32.64 %	50.24 %	53.38 %	65.44 %	79.73 %
Depth	train	63.64 %	79.61 %	81.83 %	88.80 %	97.24 %
	test	29.93 %	41.64 %	42.85 %	55.62 %	64.46 %

We may also want to ask if this architecture, when a partial input with only one modality is given, is able to infer the values of the missing modality. Several examples can be seen in Figure 9. Visually speaking, the reconstruction result seems better when both modalities are available in input even though they're both very noisy. Knowing that the two modalities are already quite similar, it reveals the difficulty of this task and the problem of multimodal retrieval might be something that is more insteresting than trying to construct information of some modality from zero.

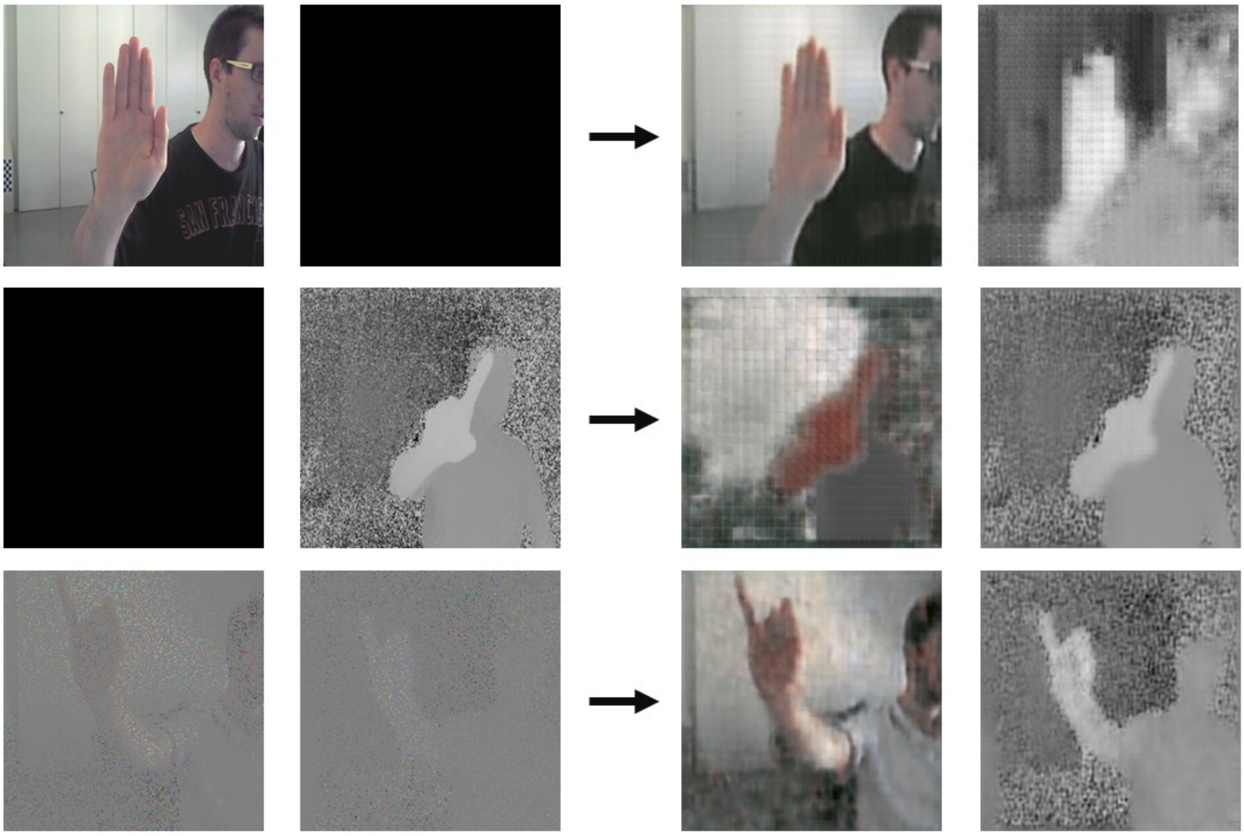

Figure 9: **Restore color and depth images from incomplete input information.**
Top) Only the color image is given.
Middle) Only the depth image is given.
Botttom) Both modalities are given but with little information (10% of pixels).

7.2 Transfer learning

In the second part of this section, we'll discuss the knowledge transfer problem between different modalities. This work was very similar to what had been done in [29], but at the same time it can also be viewed as a form of zero-shot learning [9, 41].

I studied particularly the transfer between speech and lip-reading video data using the AVLetters dataset. First the dataset was separated randomly into two parts, which I'll still call training and test set by abuse of language. They're respectively noted Tr and Te. Tr contained 600 data samples while Te comprised the left 180 instances. For X an arbitrary subset of data, X^a and X^v denote respectively the audio and video data in X.

Next, to simulate the imbalance of data quantity between different modalities in the real world, I further splitted Tr, Te intos $Tr_{A\sim T}$, $Tr_{U\sim Z}$, $Te_{A\sim T}$ and $Te_{U\sim Z}$ according to the label of each sample. For instance, a lip-reading video of a speaker saying E is in either $Tr^v_{A\sim T}$ or $Te^v_{A\sim T}$. During the first training stage, the video model was only accessible to a partial label space. The network was trained on $Tr^v_{A\sim T}$ which contained 466 samples. On the other hand, the audio model was trained on the whole label space Tr^a.

Next, I wanted to leverage speech data to fine-tune the network trained for video recognition. For a data sample $x^a \in Tr^a_{U\sim Z}$, I first computed its audio representation $h^a(x^a)$ that was taken from some hidden layer of audio model. A transfer funcion $\mathcal{T}$ was used to approximate the video representation of this data sample. That is, we want $\mathcal{T}(h^a(x^a)) \approx h^v(x^v)$. Finally, I fine-tuned the subnetwork situated after the hidden video layer that was taken as representation via a standard backpropagation algorithm.

The detailed experiement is presented in Figure 10. For both audio and video network I chose the second hidden layer for transfer. For the sake of simplicity, an instance-based approach (KNN-based mapping) was employed to define $\mathcal{T}$. I obtained mapping of a new audio input x^v by first finding the K-closest audio samples of $Tr^a_{A\sim T}$ in the representation space, and then returning the average values of the corresponding video samples (also in the representation space).

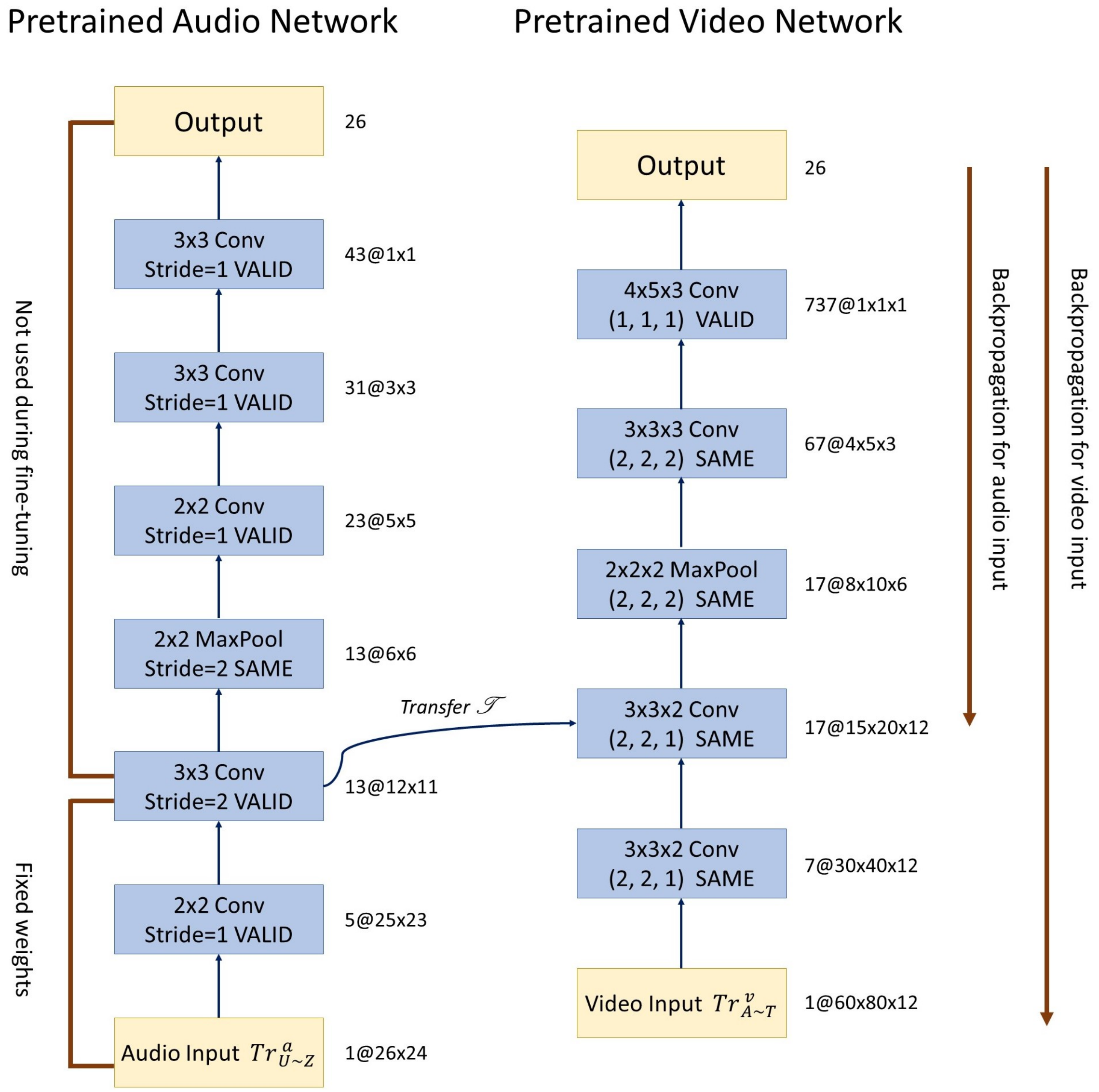

Figure 10: Illustration of the transfer learning approach applied in audio and lip-reading speech recognition tasks.
We first learn two separated model for audio and visual inputs (in my cases two CNNs) and try to fine-tune the video network with transferred audio data.

Nonetheless, if only audio inputs were presented during fine-tuning, it might cause a bias of the fine-tuned network towards the six new classes. To avoid this problem, data from $Tr^v_{A\sim T}$ were also fed as input from time to time to fine-tune the whole network. Concretely, with probability p_a audio data was presented and otherwise video input was given. At the end, the new network was tested on different parts of video data for evaluataion.

Many experiments with various hyperparameter values were conducted and some results are shown in Table 2. We see that despite the fact that the overall performance isn't improved by knowledge transfer, the fine-tuned network is now able to classify samples whose labels range from U to Z with an accuracy of 10-20% (**Exp1**). It shows the utility of the transfer learning approach although there is still a long way to go before it becomes useful in practice. During fine-tuning, the magnitude of learning rate is equally important. As we can see, if the learning rate is too high, it can deteriorate the pre-trained network and the performance may drop drastically (**Exp2**). On the other hand, as predicted before, if only audio data labeled from U to Z are provided for fine-tuining, the classification performance drops for the first twenty labels even though a higher accuracy (40-50%) can be achieved for the last six labels (**Exp3**). We also observe that there isn't a great difference when the network is tested on $Tr^v_{A\sim T}$ or $Te^v_{A\sim T}$.

Table 2: **Some results of the audio-visual transfer experiment.**
The audio model was pre-trained with $\eta = 0.1$, $\alpha_0 = 0.005$ and $\gamma = 0.8$ while the video model was pre-trained with $\eta = 0.0004$, $\alpha_0 = 0.002$ and $\gamma = 0.96$. The transfer learning experiment was carried out under the setting $\eta = 0.0004$, $\gamma = 0.8$, $K = 12$ and was trained for 160 steps. For **Exp1**, **Exp2** and **Exp3**, I used respectively $\alpha_0 = 0.001, 0.005, 0.001$ and $p_a = 0.85, 0.85, 1$.

	Tr^v	$Tr^v_{A\sim T}$	$Tr^v_{U\sim Z}$	Te^v	$Te^v_{A\sim T}$	$Te^v_{U\sim Z}$
No transfer	77.67 %	**100** %	0 %	**40.56** %	**54.48** %	0 %
Exp1	**81.17** %	98.28 %	21.64 %	39.44 %	47.76 %	15.22 %
Exp2	40.83 %	51.07 %	5.22 %	23.89 %	30.60 %	4.35 %
Exp3	19.67 %	12.23 %	**45.52** %	12.22 %	2.24 %	**41.34** %

8 Conclusion and Perspective

In this report, I have introduced two different multimodal learning settings and evaluated them on three different datasets. However, no significant results were obtained for the shared representation learning experiment and the knowledge transfer experiment aroused only moderate interest. Several possible reasons should be mentioned. First, the used datasets may be too small and monotonous and prevent the network from learning a good representation of the data. Second, since I didn't do an intense study in the domain of representation learning and dimension reduction, the algorithms applied to single modality for extracting features may not be totally adequate. Finally, the proposed experiments might be in essence inappropriate for some configurations. For example, depth maps may be redundant when color images are already present.

In spite of this, my work still ensured the possiblity and the possible benefit of taking into multiple modalities into account while dealing with real world problems. This needs to be further studied with better representation learning algorithms and bigger datasets. Ideally, we should try to apply these techniques to real data coming from human-robot interactions to see if it's possible to make the robot's behavior more humanlike thanks to multimodal fusion. On the other hand, there are still many multimodal learning models to be explored. For instance, we can project features from different sources into a joint space while trying to minimizing the distance between modalities (this is called a "coordinated representation" in [2]). I have worked a little on similar experiments but the difficulty consists in finding the appropriate loss function.

References

[1] M. Asadi-Aghbolaghi, A. Clapés, M. Bellantonio, H. J. Escalante, V. Ponce-López, X. Baró, I. Guyon, S. Kasaei, and S. Escalera. A Survey on Deep Learning Based Approaches for Action and Gesture Recognition in Image Sequences. In *Automatic Face & Gesture Recognition (FG 2017) 2017 12th IEEE International Conference*, 2017.

[2] T. Baltrusaitis, C. Ahuja and L-P. Morency. Multimodal Machine Learning: A Survey and Taxonomy. In *arXiv preprint arXiv: 1705.09406*, 2017.

[3] Y. Bengio. Practical recommondations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478, 2012.

[4] Y. Bengio, L. Yao, G. Alain and P. Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[5] C. Bregler and Y. Konig. “Eigenlips” for robust speech recognition. In *ICASSP*, 1994.

[6] A. Clark. Being there: Putting brain, body, and world together again. In *MIT press*, 1997.

[7] L. Deng, G. Hinto and B. Kinsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *ICASSP*, 2013.

[8] A. Droniou, S. Ivaldi, and O. Sigaud. A deep unsupervised network for multimodal perception, representation and classification. In *Robotics and Autonomous System*, 2014.

[9] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

[10] A. Graves. Generating sequences with recurrent neural networks. In *arXiv preprint arXiv:1308.0850*, 2013.

[11] S. Harnad. The symbol grounding problem. In *Physica D*, vol. 42, pages 335–346, 1990.

[12] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. In *IEEE Signal Processing Mag* Vol. 29, 2012.

[13] S. Ioffe and C. Szegedy. Batch normalization, accelerating deep network training by reducing internal covaraiate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[14] S. Ji, W. Xu, M Tang and K. Yu. 3D Convolutional Neural Networks for Human Action Recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, 2013.

[15] A. Karpathy, and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[16] A. K. Katsaggelos, S. Bahaadini and R. Molina. Audiovisual fusion: Challenges and new approaches. In *Proceedings of the IEEE*, vol. 103, 2015.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[19] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, vol. 86, pages 2278–2324, 1998.

[20] Y. LeCun, K. Kavukcuoglu and C. Farabet. Convolutional networks and apllications in vision. In *Circuits and Systmes (ISCAS), Proceedings of 2010 IEEE International Symposium*, pages 253–256, 2010.

[21] A. Makhzani and B. Frey, K-Sparse autoencoders. In *International Conference on Learning Representations (ICLR)*, 2014.

[22] J. Masci, U. Meier, D. C. san, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN*, 2011.

[23] I. Matthews, T. F. Cootes, J. A. Bangham and S. Cox. Extraction of visual features for lipreading. In *PAMI*, 2002.

[24] A. Memo, L. Minto and P. Zanuttigh. Exploiting Silhouette Descriptors and Synthetic Data for Hand Gesture Recognition. In *STAG: Smart Tools & Apps for Graphics*, 2015.

[25] A. Memo and P. Zanuttigh. Head-mounted gesture controlled interface for human-computer interaction. In *Multimedia Tools and Applications*, 2017.

[26] S. Mitra and T. Acharya. Gesture recognition: A survey. In *IEEE Systems, Man, and Cybernetics*, 37:311–324, 2007.

[27] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3D convolutional neural networks. In *CVPRW*, 2015.

[28] S. Molholm, W. Ritter, M. M. Murray, D. C. Javitt, C. E. Schroeder, and J. J. Foxe. Multisensory auditory–visual interactions during early sensory processing in humans: a high-density electrical mapping study. In *Cognitive brain research*, 14:115–128, 2002.

[29] S. Moon, S. Kim and H. Wang. Multimodal transfer deep learning with applications in audio-visual recognition. In *NIPS MMML Workshop*, 2015.

[30] J. Nagi, F. Ducatelle, G. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Proceedings of the IEEE International Conference on Signal and Image Processing Applications*, 2011.

[31] N. Neverova, C. Wolf, G. Paci, G. Sommavilla, G. W. Taylor,and F. Nebout. A multi-scale approach to gesture detection and recognition. In *Computer Vision Workshops (ICCVW)*, 2013.

[32] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout. Multiscale deep learning for gesture detection and localization. In *ECCVW*, 2014.

[33] J. Ngiam, A. Khosla, M. Kim, J. Nam, and A. Y. Ng. Multimodal deep learning. In: In *International Conference on Machine Learning.* 28. Bellevue, Washington, USA, 2011.

[34] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno and T. Ogata. Audio-visual speech recognition using deep learning. In *Applied Intelligence*, vol. 42, no. 4, pp. 722–737, 2014.

[35] H. Noh, S. Hong and B. Han. Learning deconvolution network for semantic segmentation. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1520–1528, 2015.

[36] L. Pigou, S. Dieleman, P. J. Kindermans, and B. Schrauwen. Sign language recognition using convolutional neural networks. In *Workshop at the European Conference on Computer Vision*, pages 572–578, 2014.

[37] G. Potamianos, C. Neti, J. Luettin and I. Matthews. Audio-visual automatic speech recognition: An overview. In *Issues in Visual and Audio-Visual Speech Processing, MIT Press*, 2004.

[38] N. Pugeault, and R. Bowden. Spelling It Out: Real-Time ASL Fingerspelling Recognition. In *Proceedings of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision*, 2011.

[39] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *arXiv preprint arXiv:1511.06434*, 2015.

[40] L. Smith and M. Gasser. The development of embodied cognition: Six lessons from babies. In *Artificial life*, 11:13–29, 2005.

[41] R. Socher, M. Ganjoo, H. Sridhar, O.Bastani, C. D. Manning, and A. Y. Ng. Zero-shot learning through cross-modal transfer. In *International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA, 2013.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, vol. 15, 2014.

[43] T. Starner, A. Pentland, and J. Weaver. Real-time american sign language recognition using desk and wearable computer based video. In *PAMI*, 20(12):1371–1375, 1998.

[44] J. Sung, I. Lenz, and A. Saxena. Deep multimodal embedding: Manipulating novel objects with point-clouds, language and trajectories. In *Robotics and Automation (ICAR), 2017 IEEE International Conference*, pages 2794–2801, 2017.

[45] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[46] C. Szegedy, S. Ioffe, V, Vanhoucke. Inception-v4, inception-ResNet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[47] V. Turchenko, E. Chalmers and A. Luczak. A deep convolutional auto-encoder with pooling – unpooling layers in Caffe. In *arXiv preprint arXiv:1701.04949*, 2017.

[48] P. Vincent, H. Larochelle, Bengio and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine Learning (ICML)*, pages 1096–1103; 2008.

[49] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P.-A. Manzagol. Stacked denoising autoencodes: Learning useful representations in a deep network with a local denoising criterion? In *The Journal of Machine Learning Research*, vol. 11, pages 3371–3408, 2010.

[50] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. In *Science*, vol. 291, no. 5504, pages 599–600, 2001.

[51] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. In *Machine Learning*, 81(1):21–35, 2010.

[52] D. Wu, L. Pigou, P.-J. Kindermans, N. D.-H. Le, L. Shao, J. Dambre, and J.-M. Odobez. Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition. In *Pattern Analysis and Machine Intelligence IEEE Transactions*, vol. 38, 2016.

[53] B. P. Yuhas, M. H. Goldstein, and T. J. Sejnowski. Integration of acoustic and visual speech signals using neuralnetworks. In *IEEE Comm. Magazine*, pp. 65–71, 1989.

[54] M. D. Zeiler, G. W. Taylor and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference*, pages 2018–2025, 2011.