Improving daily interactions of visually-impaired people: How to understand Scenes and recognize familiar Faces through a camera

Computational Vision (90539) - University of Genoa, Spring 2020

Federico Minutoli[1] 4211286 fede97.minutoli@gmail.com

Matteo Ghirardelli[1] 4147398 matteoghirardelli01@gmail.com

Sofia Bagnato[2] 4320999 sofia.bagnato1998@gmail.com

Gianvito Losapio[2] 4803867 gvlosapio@gmail.com

***Abstract*—In this report we discuss a couple of computer vision technologies which recently accomplished human-level performance on two visual tasks, namely automatic description of an image (dense image captioning) and identification of people (face identification). The implementations of such functionalities may be useful to fill some of the cognitive gaps a visually impaired person faces in his everyday life. After an overview of the state-of-the-art techniques, we present some interesting models and algorithms which could be potentially included in mobile applications: (i) Knowing When to Look (KWL) an *attention-based* captioning algorithm with deep learning, (ii) FaceNet, a *face embedding* algorithm and (iii) Sparse Support Face Machine (SSFM), a resource-efficient method for face identification. We then show some experiments which confirm their efficiency, albeit with some limitation. In the end, we discuss a possible integration with speech recognition/synthesis technologies in charge of enabling an easy interaction with the assisted person.**

***Index Terms*—Computer vision, dense image captioning, face recognition, Android, healthcare, visual-impairment**

I. Introduction

We proposed a joint project with the ambition of investigating easily accessible, state-of-the-art technologies in two different domains - Computer Vision and Speech Processing/Recognition - in order to conduct a feasibility study for the following use case:

> *"Providing a virtual assistant to visually-impaired people which can help making their daily interactions more engaging"*

More specifically, our main objective is the deployment of specific high-level cognitive functions on Android devices with low computational power.

Applications targeted for visually impaired people are unfortunately lacking in quantity, which causes some people to depend on external help for day-to-day tasks. As for the current options on the market, the most relevant products are:

- *Envision glasses*, the new AI-powered smartglasses by Envision company, empowering the blind and visually impaired to be more independent, shown in Fig. 1 [1];
- *OrCam MyEye*, a tiny device with a smart camera that can be mounted on virtually any eyeglass frame which conveys the visual information audibly, in real-time and offline [2].

Fig. 1: OrCam MyEye 2 device on a pair of glasses

This report follows our initial work on the vocal technologies - namely text-to-speech (TTS) and automatic speech recognition (ASR) - which are crucial for the success of a more ambitious application. Here we focus on two different computer vision technologies which are intended to accomplish useful visual tasks in order to assist a visually impaired person, namely dense image captioning and face identification.

The report is organized as follows. Section II is devoted to dense captioning systems and will comprise an overview of the problem, a short presentation of the state-of-the-art technologies along with a description of the algorithms we chose to use. This structure is replicated in the section III on face identification systems. Section IV is dedicated to the experiments and it is divided in 2 subsections: (IV-A) with experiments related on dense image captioning; (IV-B) with experiments related on familiar face identification. Section V contains conclusions and future works.

II. Dense captioning systems

A. Introduction

Humans can easily categorize and describe a visual scene in natural language. However, it is still a difficult task to teach to a machine. In fact, while for instance action recognition is a well-studied problem in the computer vision community,

[1] Dense image captioning [2] Face recognition

automatic understanding of complex activities and scenes is still challenging. Dense image captioning automatically generates detailed descriptions for input images. Such technology is usually referred to as "Visual to text", and has recently emerged as a prominent interdisciplinary research area, attracting more and more attention throughout the academic and industrial community. Image captioning in general can aid visually impaired users, make it easy for users to organize and navigate through large amounts of typically unstructured visual data, or make social media able to understand where we are directly from a photograph. To achieve the goal of image captioning, we have to deal with the semantic gap between low-level visual data and high-level abstract knowledge. In fact, the data form of the visual content is an image that contains a specific theme, one or more objects, a scene, an activity, etc. In contrast, the basic data form of the textual contents consists of words, which only provide high level labels of concepts and activities that abstract from the real world. In fact, all the five senses of sight, smell, hearing, taste and touch in a scene could be summarized by using a few words, whereas visual information provided by images refers only to the sense of sight. On the other hand, an image containing an activity, an object or an event is still more vivid than a set of labels. These differences are what defines the major challenges in image captioning.

The problem of image captioning is usually formulated as an end-to-end problem, meaning that the developed machine learning model should start, for instance, from raw RGB images and end up providing meaningful descriptions for these images. An approach of such kind has multiple advantages: not only it can be used to avoid to pre-process the input data for evaluation purposes and to provide a way to pack the whole model in one API ready to use by other applications, but the model itself is easier to maintain (it is frequently contained in a single framework, e.g. TensorFlow), reason with (only one set of inputs and one set of outputs to be analyzed) and it makes the optimization code easier to write (we simply tune the available parameters provided by the framework). Typically, models of such kind employ Deep Neural Networks, in which layers can specialize to perform intermediate tasks. Developing a model able to produce sentences from raw images is a quite challenging task, because image captioning connects both research fields of computer vision and natural language processing. In fact, models relying only on features extracted with CNNs may not be powerful enough to extract the information which is usually conveyed by high level abstract knowledge, unless the images themselves embed pieces of the image description. On the other hand, recursive neural networks (i.e., LSTMs) are usually used for the purpose of language modeling and word prediction in NLP and NLG, and are useful when trained on image annotations for captioning purposes, but may not take into account whether the generated words are correlated with the visual features produced by CNNs. In order to solve these limitations, and therefore to provide models able to implement feedback between the visual context and the language context, various hybrid approaches have been proposed by researchers, including the ones based on the Encoder-Decoder framework. The main aim of our project is to investigate the nature of a subclass of these methods, which is the one comprising the so-called Attention-based methods. In particular, we will study the architecture of an Adaptive Attention-based method for image captioning and its main implementation details. We will then train the associated model on a cluster of machines, in order to later test its performance using various metrics tailored for the image captioning problem. We will then show how the model works in the evaluation step, by providing significant plots that make use of the so-called visual grounding probabilities and attention weights. We will then deploy the model on an Android application, in order to integrate the image captioning functionality with a TTS engine; by doing this way, we aim to perform a step forward in building an application to interactively help visually impaired people, whose baseline has already been developed in the following repository: https://github.com/DiTo97/stunning-potato.

B. State of the art

The earliest approaches in addressing the image captioning problem are the ones based on language rules and template-based generation of words. For instance, H. Nagel (1988) [18] proposed a model capable of filling the key elements in a sentence with the information provided by detected bounding boxes. This was achieved by using the concept of case grammar, which focuses on the link between the valence, the number of subjects, objects, etc., of a verb and the grammatical context it requires. Later approaches developed by M. Khan and Y. Gotoh (2012) [19] required to use a context free grammar in order to map a limited set of recognized human activities to a single meaningful sentence. Other methods proposed by Yang et. al (2011) [20] had the aim of adding prepositions and activities based on a language model learned from a text corpus, for instance a Hidden Markov Model (HMM). The main drawback of language rules and template-based generation is that they expect the intra-class variance of each visual content set to be small, and therefore they are effective in a limited domain.

Another category of methods used for image captioning has the main aim of reducing the sentence generation process to borrowing available descriptions from other visual contents. For example, Ordonez et. al (2011) [21] proposed a method which searched for the most relevant captions in a dataset by applying a greedy search strategy. The method used to evaluate the similarity between sentences was, in this case, a sentence-semantic representation mapping. Mason et. al. (2014) [22], instead, presented the image captioning problem as fully data-driven by incorporating visual and textual features in a single bag-of-words model. All the approaches based on querying existing sets of visual contents achieve good performance when the images used in the training set are related to the ones contained in the test set in some way, otherwise the model performance results in being significantly degraded. Again, this class of methods is effective in a limited domain.

A first step towards the Attention-based Encoder-Decoder methods cited in the previous section was made by image captioning techniques which incorporated descriptions into visual to text models after visual recognition. The methods comprised in this group rely on both textual data and visual data: they often represent text and images jointly in a latent space, and then find the matched counterpart for a given query, which can be either image-based or sentence-based. An example of this approach is the one presented by Socher et. al. (2014) [23], which uses a recursive neural network based on dependency trees.

Further advances in solving the image captioning problem have been made by employing language models to directly generate textual descriptions to be associated to images. For instance, Yao et al. (2010) [24] developed a NLG (natural language generation) method that converts image parsing results to textual descriptions. This method used four components: an image parsing engine, a graph-based visual knowledge representation (which incorporates Entity-Relation-Entity (E-R-E) and Entity-Attribute-Value (E-A-V) information contained in the image) a semantic web (based on meta-information associated to Internet-based resources and on the ability to reason about such information) and a text generation engine. Some other works focus on training a language generation model on a separate aligned corpus of images with descriptions, in order to later integrate this model with more advanced language generation methods (Kuznetsova et al., 2012 [25]). The main challenge in developing methods belonging to this last classical category of models consists in how to effectively adapt the language models to the tasks of *"Visual to text"*. In fact, most existing works explore the language models after fixing the visual learning model, but it would be more effective to jointly train the visual and language models.

The state of the art models used nowadays for image captioning are based on deep learning. The first pioneering works that introduced deep neural networks in image captioning shared similar ideas with previous methods, like the already introduced framework developed by Socker et. al. (2014) [23]. In particular, Kiros et. al (2015) [26] considered image features encoded with a CNN as the bias within a log-bilinear model (LBL, usually a feed-forward neural network with a single linear hidden layer), which is in charge of associating a sentence to the input image. Other works from Karpathy and Li (2015) [27] explored the alignment between image regions and captions, by combining a Region-CNN with a bidirectional RNN model. In these early works, it is noticeable the intention of the researchers to orient the structure of the developed methods to the adoption of the Encoder-Decoder framework, which is depicted in **??**.

This framework is made, in general, by two main components:

1) A pre-trained CNN (usually on a large-scale dataset), which extracts various vision cues from one still image as a single real-valued feature representation.
2) A recurrent neural network (RNN), which is in charge of generating the caption for the input image conditioned on the features produced by the CNN.

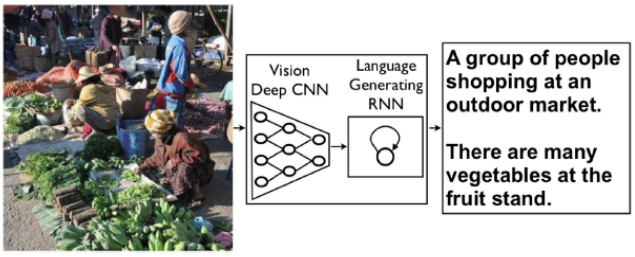

Fig. 2: General structure of the Encoder-Decoder framework

In particular, the aim of the Encoder-Decoder framework is to maximize the following objective function:

$$\theta^* = \arg\max_{\theta} \sum_{(I,y)} \log(p(y|I;\theta)) \tag{1}$$

where the pair (I, y) represents an image and the corresponding correct transcription of a caption (in other words, the caption as reported in the training set). Therefore we optimize the sum of the log probabilities of predicting the correct caption given each image of the training set by means of stochastic gradient descent (i.e., Adam optimizer). Hence, this is a maximum log-likelihood problem, aimed at inferring the best configuration θ of the model's parameters.

Each log probability $\log(p(y|I;\theta))$ can be decomposed using the following formula:

$$\log(p(y|I)) = \sum_{t=1}^{T} \log(p(y_t|y_{t-1}, y_{t-2}, ..., y_1|I)) \tag{2}$$

where we dropped the dependency on θ for convenience. The conditional probability $\log(p(y_t|y_{t-1}, y_{t-2}, ..., y_1|I))$ of generating a particular word given the previous words is modeled by using the RNN, in the following way:

$$\log(p(y_t|y_{t-1}, y_{t-2}, ..., y_1, I)) = f(h_t, c_t) \tag{3}$$

where f is a non-linear function that outputs the conditional probability of y_t, c_t is the visual context vector at time t extracted from the image I and h_t is the hidden state of the RNN at time t. As we will see later, the context vector is typically computed as a weighted sum of feature vectors.

Instead of a vanilla RNN, state of the art models usually employ a LSTM, because it has shown state-of-the art performance on sequence tasks such as the already mentioned translation between languages. The memory cells of the LSTM networks can in fact maintain information in memory for long periods of time. Moreover, it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies, like the one of image captioning. We recall that a generic LSTM computes, at time step t, the hidden state h_t in the following way:

$$h_t = LSTM(x_t, h_{t-1}, m_{t-1}) \tag{4}$$

where x_t is an input vector, h_{t-1} is the hidden state of the LSTM at time $t-1$, m_{t-1} is the memory of the LSTM at time $t-1$. When using a LSTM for the purpose of image captioning, it is important to define how the input vector x_t will be formed. Typically, current approaches try to build a single vector comprising of both image features and words embedding associated to that feature.

An example of implementation of the Encoder-Decoder framework for image captioning is Google NIC, proposed by Vinyals et al (2015) [28]. The general structure of this model is depicted in Fig. 3 below.

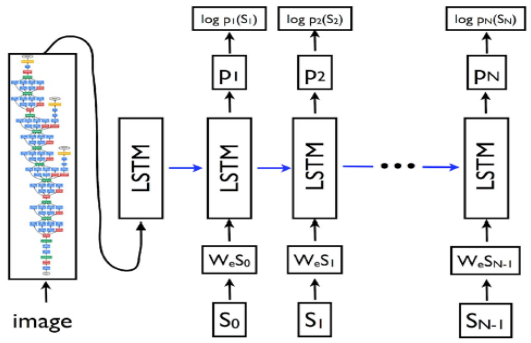

Fig. 3: Google NIC

NIC unrolled the procedure of caption generation in the following sequence of steps:

1) $x_{-1} = W_I \cdot CNN(I)$
2) $x_t = W_e S_t$
3) $h_{t+1} = LSTM(x_{t+1}, h_t, m_t)$
4) $p_{t+1} = softmax(h_{t+1})$

where W_I and W_e are, respectively, an image embedding matrix and a word embedding matrix that need to be learned by the model, and p_{t+1} is the probability distribution over all the possible words of a pre-defined vocabulary, that is, the score for each word of being selected at time t. Step (1) means that NIC only shows the image content to the RNN decoder once (at time $t = -1$). A newer version of NIC was released in 2019 with a few improvements over the original proposition. More specifically, they substituted the original CNN encoder with CutMix [29], which has in turn established it as the current state of the art image captioning model.

Many other works proposed in 2015 share the same architecture as NIC, with subtle differences. For instance, Mao et. al. [30] proposed a multimodal RNN instead of a vanilla RNN. In this case, the image content is visible to the RNN at each time step, and visual as well as text information are incorporated together before applying the softmax function to predict words. Chen et al. [31], instead, introduced a recurrent visual feature to assist the long-term memory for the RNN decoder. In both these works, it is already evident the intention of the researchers to push the state of the art towards *attention-based* models, which are able to determine whether regions of an image are visually significant (spatially and semantically) for the corresponding word predictions.

Attention-based image captioning differs from the original Encoder-Decoder framework in that it allows the decoder to look at different image regions according to the specific context considered at each time step. The attention mechanism is widely used nowadays for image captioning, and allows the decoder to learn where and what it should attend to in order to generate words. Therefore, unlike vanilla Encoder-Decoder methods, attention-based techniques don't keep the image content fixed during during natural language generation. Attention methods can be categorized as follows:

1) *Spatial attention methods*
2) *Semantic attention methods*

The first implementation of category (1) was introduced by Xu et al. (2015) [32]. This implementation modeled the last convolutional layer of a pre-trained CNN as the image encoder, ending up with visual information vectorized as a set of representations $A = \{a_1, ..., a_L\}, a_i \in R^D$, where L is the number of regions of the given image. This mechanism allows the RNN decoder (in this case, formulated as a single layer LSTM) to attend to different spatial regions when generating the next words. In particular, the visual context vector c_t used to model the aforementioned probability $f(h_t, c_t)$ is not steadily single-based on image features produced by the CNN, but is instead re-computed at each time step by assigning a weight to each image region. The weights assigned to each region are computed by using a specific attention function.

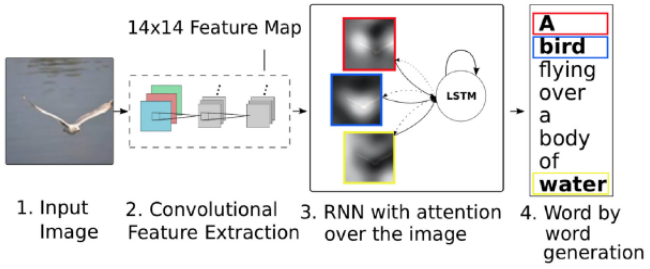

Fig. 4: Structure of a spatial Attention-based image captioning model, as introduced by Xu et al. (2015)

The second category of methods was first introduced by You et al. (2016) [33]. They developed a model which shifted the attention mechanism from spatial image regions to a concept of "*visual words*". In their model, they only show image features at the beginning to give an overall visual clue to the RNN decoder, and then design two attention models to be used at each time step that take care of input and output of the decoding process. More in detail, the input attention model takes as input a set of visual attributes (i.e., semantic information represented by words embedding) detected from an image (usually by using an object detector), and learns a semantic context vector which is conditioned on the previous

word. On the other hand, the output attention model adopts a similar attention mechanism to the input one, except for being conditioned on the current hidden state.

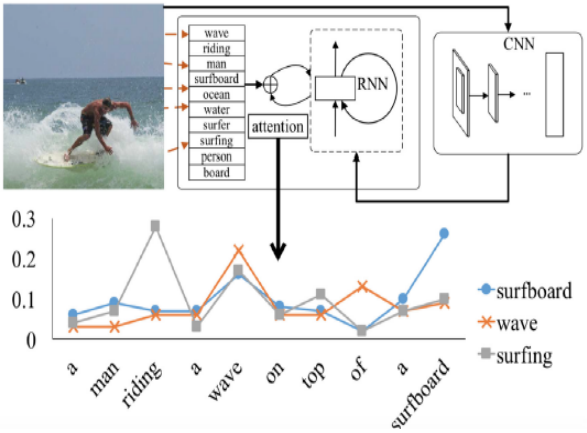

Fig. 5: Structure of semantic Attention-based image captioning model, as introduced by Xu et al. (2015)

In the next section, we will consider spatial Attention-based models, and in particular, we will present the method chosen for this project, which uses an Adaptive spatial attention mechanism used to generate captions for input images.
The method was introduced in the paper "Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning" Lu et al., 2017) [34].

C. KWL

The method proposed by Lu et al. (2017) [34] follows exactly the architecture of the spatial Attention-based Encoder-Decoder framework already described in the previous section, with some differences. Indeed they noticed that not every generated word carries visual information (like conjunctions, attributes or words frequently found in noun phrases) which, as they hypothesized, could actually disturb the model, in a sort of desperate hunt of a visual counterpart. Hence, they proposed an extension to the classic LSTM model comprising a fourth gate, that they referred to as "*sentinel gate*", which should have played the role of guidance to the model in either attending to the image (and, if so, to which regions) or to the visual sentinel vector. Belonging to the family of the Encoder-Decoder framework, the first choice falls down to which CNN encoder to use. They relied on a pre-trained ResNet-152, a well-known object detector, as they were interested in the spatial features representative of the output of the last convolutional layer that ResNet would generate from an image. These features have dimension $2048 \times 7 \times 7$, where 7×7 is the shape of a single feature. Therefore we end up with a matrix $A = [a_1, ..., a_k]$, with $k = 49$, that is, the unrolled 7×7. Each vector $a_i \in R^{2048}$ is the representation of all the spatial image features for a certain grid location. A global image feature $a_g \in R^{2048}$ is then obtained by computing the 2D average pooling on each 7×7 matrix.

Two single layer perceptrons and a ReLU activation function are then used to transform each vector a_i and the global feature vector a_g respectively into embedded vectors of visual features $v_i, v_g \in R^d$, where d represents the dimensionality of the embedding space of choice [1].

$$v_i = ReLU(W_a a_i) \tag{5}$$

$$v_g = ReLU(W_b a_g) \tag{6}$$

where W_a and W_b are weight parameters to be learnt. The embedded vectors v_i will be stored in a matrix $V = [v_1, ..., v_k]$, where $V \in R^{d \times k}$.

The next step is to initialize the first predicted word of the LSTM, with the tag "*start*", marking the beginning of a sentence generation. To do so, it is necessary to convert the aforementioned tag into the corresponding word embedding w_t, after fixing a big enough embedding size (i.e., $d/2$ mentioned in the footnote), where each word in the vocabulary can be mapped to a unique vector.

We then concatenate the global image feature v_g with w_t in order to produce $x_t = [w_t; v_g]$. This vector x_t will be the input for an extended version of the LSTM, which performs the following computations:

$$g_t = \sigma(W_x x_t + W_h h_{t-1}) \tag{7}$$

$$s_t = g_t \odot tanh(m_t) \tag{8}$$

where W_x and W_h are weight parameters to be learned, g_t is a LSTM forget gate applied to the memory cell m_t, $\odot$ represents the element-wise product, σ is the logistic sigmoid activation function and s_t the proposed "*visual sentinel*".
This component was introduced by Lu et. al in the adaptive Attention-based framework as a more powerful form of visual memory, through which the model remembers what it has already generated (and, in turn, which image regions it has already attended to). In order to do so, the visual sentinel, that from here on we will call s_t, extends the fully visual context vector, typical of spatial attention-based RNNs. In other words, if the currently examined image region is not visually significant for modeling the prediction of the next word then the model will solely rely on the visual and linguistic context it has stored up to step $t - 1$.

The visual context at time step t is computed as a context vector c_t, which is defined in the following way:

$$c_t = g(V, h_t) \tag{9}$$

where g is the attention function, V the aforementioned spatial features and h_t the hidden state of the LSTM at t.
This formulation of the context vector follows the family of Luong attention systems were the context is computed on the current hidden state from the decoder, which was, again, proposed by the authors, as opposed to the more classical

[1] In the code you would actually see v_i vectors $\in R^d$, also equal to the hidden size of the LSTM, whereas the embedding size of v_g would be $d/2$ to make it compliant with the words embedding vector w_t.

formulation of Bahdanau attention systems, where the context would be inferred by the previous hidden state h_{t-1}.

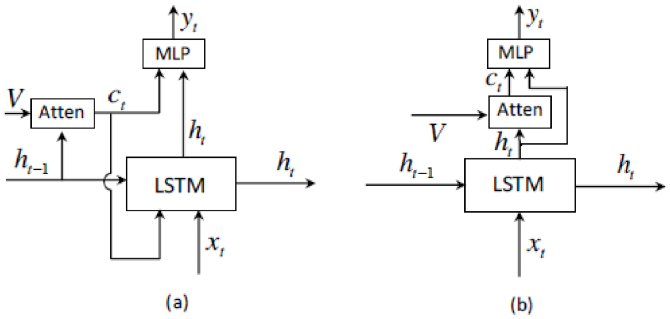

Fig. 6: A illustration of soft attention model (a) and the proposed spatial attention model (b).

The classical attention map α_t would be obtained by feeding the visual features V and the hidden state h_t to a single layer perceptron with $tanh$ as activation function, with the aim of gathering a vector of k scores, one per region, (turned into a distribution through a $softmax$ layer) indicating the most relevant region for the subsequent step $t+1$:

$$z_t = w_h^T tanh(W_v V + (W_g h_t)1^T) \tag{10}$$

$$\alpha_t = softmax(z_t) \tag{11}$$

where w_h, W_v and W_g are all weights to be learned with dimensions coherent to produce a final output α_t in R^k and 1^T is a matrix full of 1s. α_t is decisive in the choice of regions, as a common greedy computation of the visual context c_t is expressed as a simple linear combination of the attention scores with the visual features v_i.

$$c_t = \sum_{i=1}^{k} \alpha_{ti} v_{ti} \tag{12}$$

What the authors proposed was to insert s_t within the set of scores α_t by adding one more channel to the vector through concatenation, effectively interpreting this $k+1$-th score as β_t, that is, the sentinel gate regulating the fidelity of the sentinel. So, they extended (11) turning it into:

$$\hat{\alpha}_t = softmax([z_t; w_h^T tanh(W_s s_t + (W_g h_t))]) \tag{13}$$

with $\hat{\alpha}_t$ in R^{k+1}, where the last term directly depends on s_t, $\beta_t = \hat{\alpha}_t[k+1]$, as Fig. 7 illustrates.

After extracting the sentinel gate β_t, the context-aware vector $\hat{c}_t$ - aware meaning that it's aware of the value of the visual sentinel s_t - can be computed as such:

$$\hat{c}_t = \beta_t s_t + (1 - \beta_t) c_t \tag{14}$$

where, depending on the value of β_t the model could solely focus on the visual sentinel ($\beta_t = 1$) or on the visual features ($\beta_t = 0$), as it did before, much like how rates that bridge multi-task problems together function.

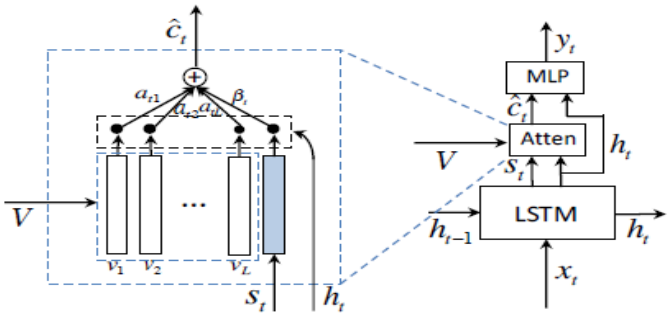

Fig. 7: An illustration of the proposed model generating the t-th word y_t given the image with the introduction of s_t.

At this point, a simple function can be implied to compute the log-likelihood of words ($f(c_t, h_t)$) over the vocabulary $\mathcal{W}$, that takes all the above into account:

$$p_t = softmax(W_p(\hat{c}_t + h_t)) \tag{15}$$

where W_p is a weight to be learnt that maps the vectors from the hidden space of the LSTM in R^d to the underlying vocabulary space in R^j with $j = |\mathcal{W}|$.

D. Implementation details

Setting up a functional implementation of the KWL model was not an easy feat, as many components play a role as we've seen in the previous subsection. Fortunately enough, there are quite a few resources online [37]–[41] that try to bridge the gap between the mathematical definition and the actual code, making use of the most common deep learning Python libraries (i.e., TensorFlow, PyTorch, etc...). We've separated concerns in our repository between code related to pre-processing (summarized within *preprocessing.sh* as a one-for-all script to set everything up) and actual code. The most important modules are, in order, *build_vocab.py*, *utils.py*, *data_loaders.py* and *models.py*, whose names are pretty self-explanatory, which are also needed to perform any experiment in the notebook. The main entry point to the model is, of course, self-contained within *models.py* where a wrapper to the whole architecture, which goes by the name *Encoder2Decoder*, handles both training and sampling phases of the process, with independent modules for the ResNet-derived encoder and the LSTM decoder with its independent attention and sentinel blocks.

We've chosen to tailor the model with the SPICE [17] metric (refer to *Experiments* section for more details on this part), used as an early-stopping validation condition, and we've applied Standford's PTB tokenizer [36] to lemmatize captions both in training and evaluation. Despite PTB being a really efficient implementation of a tokenizer, the lemmatization procedure is a hard bottleneck to both the creation of the vocabulary (6-hour-long process) and evaluation across epochs (50 minutes per-epoch), which has given us many troubles in

training the model on Colab first, and on CERN's HTCondor cluster later, due to their hard cap of 12 hours on active runtime that weren't enough for COCO's 5GB training dataset.

We've chosen Adam as the optimizer of choice to perform stochastic gradient descent, both for the overall architecture and for the first 5 layers of the CNN encoder only. Indeed many resources suggested to fine-tune the higher layers of the CNN independently from the rest of the model (as it is already pre-trained on a huge dataset, like ImageNet) and to do that only after the architecture has gone through a good number of training epochs on its own. In our case, this number was fixed to 20 epochs, which also coincides with the epoch after which the learning rate (fixed at 4e-4 for the architecture and 1e-4 for the CNN layers, as they come in play later) starts decaying exponentially every other 20 iterations.

Regarding the sizes of the word (feature) embedding space and LSTM's hidden space we sticked to the values suggested by the authors of 256 and 512, respectively. Also, we implied a batch size of 64 to avoid stressing the GPU's memory too much under intensive workload. Finally, as the *exploding gradients* is a problem that plagues RNNs especially (which is also true for weight matrices W values), due to the length of the sequences they deal with often crossing the 30 mark, we clipped the gradient at 0.1 after every epoch [42], [43].

- Drive repository

III. Face identification systems

A. Introduction

In the last few years, applications being devoted to identify people in images moved from the realm of science fiction into daily life. This is possible thanks to the recent advances in face recognition systems. According to the recent *Facial Recognition and Biometric Technology Moratorium Act* [3], "the term *facial recognition* means an automated or semi-automated process that (A) assists in identifying an individual, capturing information about an individual, or otherwise generating or assisting in generating surveillance information about an individual based on the physical characteristics of the individual's face; or (B) logs characteristics of an individual's face, head, or body to infer emotion, associations, activities, or the location of an individual."

The bill was introduced last June, 2020 by US Democratic lawmakers with the purpose of banning the use of facial recognition technology by federal law enforcement agencies. The proposed law has arrived at a point when the police use of facial recognition technology is coming under increased scrutiny amid protests after the killing of George Floyd in late May. Studies have repeatedly shown that the technology is less accurate for Black people [4], and also the New York Times recently reported that an innocent Black man in Michigan was arrested after being misidentified by facial recognition software [5].

As a consequence, IBM, Amazon, and Microsoft said they would stop selling facial recognition software to law enforcement. IBM's policy is in effect indefinitely, while Amazon and Microsoft said theirs are temporary [6].

Coming to the realm of computer vision, similar problems and commercial solutions related to faces in images exist which may be combined or mapped to each other. In this project, the focus will be on face identification and on face verification.

More specifically, face identification can be formalized as follows. Given a dataset $\mathcal{D} = \{V, Y\}$ - where $V = (v_i)_{i=1}^n$ is a set of face images of size $h \times w \times r$, namely $v_i \in \mathbb{R}^{h \times w \times r}$, and $Y = (c_i)_{i=1}^n$ are labels denoting the corresponding identity with $c_i \in C = \{c_k\}_{k=1}^p$ - we would like to find a classifier $f : \mathbb{R}^{h \times w \times r} \rightarrow C$ able to correctly match each face to its identity on $\mathcal{D}$ and on future data.

It is worth noting that: (i) the adjective *familiar* can be used to suggest a subjective perspective on the set of identities C, for instance the familiar faces a visual impaired person may be interested in recognize; (ii) a special label in C can be used to denote all the possible "unknown" faces.

Formally, face verification only differs from face identification in that the problem is split in many one-vs-all binary classifications, i.e. we may separately want to verify each identity c_k in the set C against all the remaining identities $C \setminus \{c_k\}$ (one "genuine" user versus any eventual "impostor" attempting to break a security system). Different emphasis is put on metrics, as will be explained in the section IV-B.

In the next section III-B we will start with presenting a brief overview of the current state of the art methodologies, then explaining more in depth why and how deep learning is used to extract features. After this theoretical introduction we will discuss the FaceNet method in section III-C, the one among the two we chose to study that is based on deep learning, and the Sparse Support Face Machine (SSFM) in section III-D.

As it will become clear, the main reason behind their choice is that they can be potentially combined to identify faces starting from very few images collected as input. Moreover, they show efficient computational complexity and can be easily employed to design mobile applications.

B. State of the art

Modern state-of-the-art methods rely on deep learning in order to boost performance. The complex structure of DNNs, along with the increasing size of the publicly available datasets, has been the reason for modern advances in state-of-the-art methods and results.

Up until 2011 the image recognition pipelines relied on hand-engineered features (e.g. SIFT or Haar). This started to change around 2010; for example, a paper to introduce a change was *Building High-level Features Using Large Scale Unsupervised Learning*, by Le et al, where the image features were learned in an unsupervised setting by training an autoencoder in order to minimize the reconstruction error; subsequent works and advances paved the way to, for example, DeepFace and FaceNet models.

1) Deep learning methods: Using custom built features tends to result in over specialized feaure extractors, thus making them difficult to use in different domains. This problem is easily solved by employing feature *learning* methods, whose

learnt feature extractors have been successfully used among many different domains.

The general structure of a feature learning algorithm is the following:

1) Create a structure with parameters θ
2) Define a loss function L
3) Minimize L with respect to θ

The loss minimization is usually achieved with gradient-based optimization algorithms.

There are two main categories among deep learning methods, which are *supervised* and *unsupervised* ones. An example of an unsupervised feature learning algorithm is an autoencoder (such as the one presented by Le et al), which is formally composed by two parts, an encoder and a decoder. The encoder function is parametrized by θ and f_θ and the decoder one by $\theta^{'}$ and $g_{\theta'}$; loss can either be squared error or cross-entropy. In addition to being used as feature extractors, autoencoders can also be used to pre-train a DNN. As inputs to autoencoders and DNNs are flattened the local image structure is lost; a way to solve the problem is employing a convolutional neural network (CNN)

An example of a supervised feature learning algorithm is Google's Facenet, which employs a loss based on the distance between similar examples' features in the output space; it will be discussed in the next section.

C. FaceNet

FaceNet is a method proposed by Florian Schroff *et al* in 2015 [9] which can be used for face verification, recognition and clustering which is based on learning a Euclidean embedding per image using a deep convolutional network. The network outputs live in an embedding space where the vector distance is directly proportional to the face similarity, i.e. vectors obtained from images of faces belonging to the same person lay near to each other in the embedding space.

Once the feature extractor is trained, the problems mentioned before (verification, recognition and clustering) become trivial as they can all be solved with simple classifiers or methods based on vector distance.

These vectors belong to $\mathbb{R}^{128}$ and the network is trained by using a triplet loss which, given an *anchor*, minimizes the distance between it and a positive (i.e. a sample having the same identity) and maximizes the distance between it and a negative.

1) Triplet- loss and selection: Given the aforementioned embedding space $\mathbb{R}^{128}$, training our embedding network can be seen as learning a mapping function $f(x) : X \to \mathbb{R}^{128}$, where X represents the image space. An additional rule is that this embedding must live in a d-dimensional hypershpere (with d=128), a rule that is enforced by doing the l2-normalization of the network output. The triplet loss is described by the following formula:

$$L = \sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{16}$$

where x^a represents the *anchor*, x^p a positive example, and x^n a negative one.

In order to properly train the network it is important to choose suitable triplets (x^a, x^p, x^n). Since fast convergence is a desiderable property it is important to choose hard triplets, i.e. given a x_i^a we want to select a *hard positive* and a *hard negative* such that

$$\begin{cases} x^p = argmax_{x_i^p} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 \right] \\ x^n = argmin_{x_i^n} \left[\|f(x_i^a) - f(x_i^n)\|_2^2 \right] \end{cases} \tag{17}$$

This presents a few challenges: not only computing the argmin and argmax across the whole training set is really expensive, but choosing only triplets satisfying these constraints would result in poor training, as we would only select mislabelled and low quality images.

The solution explored in the paper is using large mini-batches (a few thousands exemplars) and selecting all the positive and the hard negative exemplars from within a mini-batch. Regarding the negatives, specifically all the negatives satisfying $\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$ are selected (they're called *semi-hard negatives*), whereas in order to have a meaningful representation of the anchor-positive distances Schroff *et al* sampled the training data such that about 40 faces were selected per identity in each mini-batch.

2) Training and network selection: The CNN is trained with stochastic gradient descent with standard backprop and AdaGrad optimization and $\alpha = 0.2$. In most of the experiments the learning rate starts at 0.05 and is then lowered to finalize the model. In the original paper two different architectures for the CNN are considered: one is the Zeiler & Fergus network, the other (also used by the implementation of our choice) is the Google Inception ResNet.

They differ in the number of parameters and FLOPS (Floating Point Operations Per Second; metric used to to describe how many operations are required to run a single instance of a given model), with GoogLeNet having 20X less parameters and up to 5X less FLOPS, depending on the specific Zeiler & Fergus and Inception architectures considered.

3) Original Paper Results: In the original paper the FaceNet method is evaluated on four different datasets; with the exception of the LFW dataset (labeled faces in the wild) and the Youtube Faces dataset, the method was evaluated on the task of face verification.

As for the results, the validation accuracy increases with the FLOPS, thus there is a trade-off between the model complexity and the obtained accuracy; fig. 8 shows the graph the authors included in their paper, which considers various architectures.

D. Sparse Support Face Machine

Sparse Support Face Machine (SSFM) is a face verification algorithm originally proposed by Biggio *et. al* in 2015 [7]. A coincise description follows. Hereafter, superscripts are used to emphasize the dependence on the choice of one "genuine" identity c_k in order to define a specific binary classification problem.

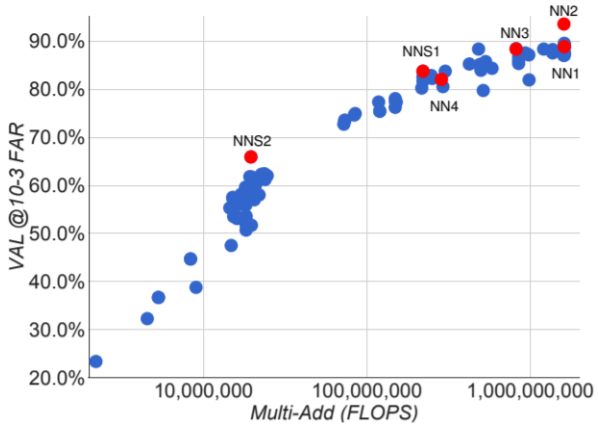

Fig. 8: Observed FLOPS vs. accuracy trade-off. All the circles represent one possible architecture; the four red ones represent the ones the authors focused on during their experiments.

The first step is to apply a transformation $\Phi : \mathbb{R}^{h\times w\times r} \to \mathbb{R}^d$ to the original images

$$V = (v_i)_{i=1}^n \xrightarrow{\Phi} X = (x_i)_{i=1}^n \tag{18}$$

in order to extract meaningful features and at the same time reduce the dimensionality of the input data, , namely $d \ll h \times w \times r$.

For each identity $c_k \in C$ a set of binary labels $Y^{c_k} = (y_i^{c_k})_{i=1}^n$ is associated to the features $(x_i)_{i=1}^n$

$$y_i^{c_k} = \begin{cases} +1 & \text{if} \quad c_i = c_k \\ -1 & \text{otherwise} \end{cases} \qquad i = 1, \dots, n \tag{19}$$

and a binary support vector machine classifier (SVM) is trained on $\mathcal{D}^{c_k} = \{X, Y^{c_k}\}$ to learn a discriminant function $g_{\text{SVM}} : \mathbb{R}^d \to \mathbb{R}$ of the form

$$g_{\text{SVM}}(x; \alpha, b) = \sum_{i=1}^n \alpha_i \, y_i^{c_k} \, s(x, x_i) + b \tag{20}$$

where:

- $\alpha = [\alpha_1, \dots, \alpha_n]^\top$ are the coefficients and b is the offset term used to define the separating hyperplane;
- $s(x, x_i)$ is an appropriate kernel called *matching score* which quantifies the similarity between two feature vectors (ideally in terms of identities). In the following, the radial basis function is considered:

$$s(x, x_i) = e^{-\gamma \|x - x_i\|^2} . \tag{21}$$

In particular, the discriminant function of the identity c_k, denoted by $g_{\text{SVM}}^{c_k}$, is obtained by minimizing the SVM dual problem objective

$$\begin{aligned} (\alpha^{c_k}, b^{c_k}) \leftarrow \arg\min_{\alpha} \; & \alpha^\top Q \alpha - e^\top \alpha \\ \text{subject to} \quad & \alpha^\top Y^{c_k} = 0 \\ & 0 \le \alpha_i \le C u^{c_k} , \; i = 1, \dots, n \end{aligned} \tag{22}$$

where

- Q is the $n \times n$ matrix containing matching scores of the training sample, namely $Q_{ij} = y_i y_j s(x_i, x_j)$;
- C is the regularization parameter with u_{c_k} acting as a weight used to compensate imbalance of the positive class c_k with respect to the negative one.

The function $g_{\text{SVM}}^{c_k}$ is later thresholded to perform classification:

$$f^{c_k}(x) = \begin{cases} +1 & \text{(identity } c_k \text{ verified)} \\ -1 & \text{(identity } c_k \text{ not verified)} \end{cases} \quad \begin{array}{l} \text{if} \quad g_{\text{SVM}}^{c_k}(x) \ge \theta^{c_k} \\ \text{otherwise} \end{array} \tag{23}$$

where θ^{c_k} is an empirical user-specific threshold.

It is worth noting that typically only few coefficients α_i in the solution are not close to zero. As a consequence, this approach enables verifying a claimed identity through a linear combination of matching scores computed between a given face x and a subset of face templates called *support faces* $S_f^{c_k} = \{x_i \in \mathcal{D}^{c_k} \,|\, \alpha_i \neq 0\}$ automatically selected by the SVM learning algorithm to discriminate the training data.

The novel formulation of the paper is to replace the discriminant function $g_{\text{SVM}}^{c_k}$ with the following approximation:

$$h(x; \beta, z) = \sum_{j=1}^m \beta_j \, s(x, z_j) \tag{24}$$

where:

- $z = \begin{bmatrix} z_1 \dots z_m \end{bmatrix}^\top$ with $z_j \in \mathbb{R}^d$ is the $m \times d$ matrix of the so-called *virtual support faces* obtained through a principled transformation of the set $S_f^{c_k}$;
- $\beta = \begin{bmatrix} \beta_1 \dots \beta_m \end{bmatrix}^\top$ with $\beta_j \in \mathbb{R}$ is the $m \times 1$ vector of the corresponding coefficients.

Such approximation is computed by solving the following minimization problem

$$\beta^{c_k}, z^{c_k} = \underset{\beta \in \mathbb{R}^m, \, z \in \mathbb{R}^{m \times d}}{\arg\min} \; \Omega(\beta, z; \mathcal{D}^{c_k}) \tag{25}$$

with

$$\begin{aligned} &\Omega(\mathcal{D}^{c_k}; \beta, z) = \\ &= \frac{1}{n} \sum_{i=1}^n u_{y_i^{c_k}} \left(h(x_i; \beta, z) - g_{\text{SVM}}^{c_k}(x_i) \right)^2 + \lambda \, \|\beta\|^2 \\ &= \frac{1}{n} \left(\beta^\top S_{xz}^\top U \beta S_{xz} - 2 \beta^\top S_{xz}^\top U g + g^\top U g \right) + \lambda \|\beta\|^2 \\ &= \frac{1}{n} \left(h^\top U h - 2 h^\top U g + g^\top U g \right) + \lambda \|\beta\|^2 \end{aligned} \tag{26}$$

where

- S_{xz} is the $n \times m$ similarity matrix containing matching scores between training points and virtual support faces, namely $S_{xz}(i, j) = s(x_i, z_j) = e^{-\gamma \|x_i - z_j\|^2}$;
- U is the $n \times n$ diagonal matrix containing the weights u^{c_k} for the identity c_k;
- g is the $n \times 1$ vector containing the learned SVM discriminant function computed on the training points X;

- $h = S_{xz}\beta$ is the $n \times 1$ vector containing the learned approximation function computed on the training points X.

This is a *reduction method*: the aim is to minimize the distance on the training points between the target function $g_{\text{SVM}}^{c_k}$ and and its approximation h, with respect both to β and to the choice of the subset of the support faces z. In order to attain a better trade-off between accuracy and number of support faces, z is not constrained to belong to $\mathcal{D}$ but the algorithm is allowed to create new, *virtual* faces in $\mathbb{R}^d$. An intuitive representation is shown in fig. 9.

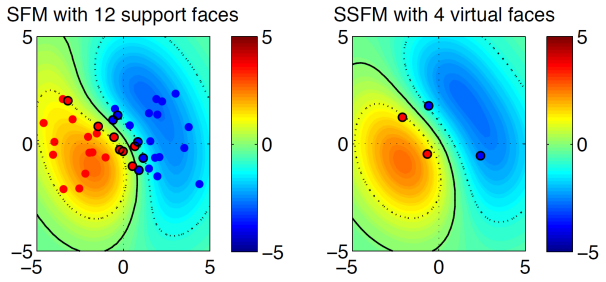

Fig. 9: A two-dimensional example taken from the original paper [7]. A colormap represent the values of the discriminant function $g_{\text{SVM}}(x;\alpha,b)$ for an SFM (left) trained on 40 samples (red and blue points in the left plot), and of $h(x;\beta,z)$ for the corresponding reduced SSFM (right). Note how $h(x;\beta,z)$ almost exactly replicates the SFM's decision boundary $g_{\text{SVM}}(x;\alpha,b) = 0$ (solid black line) using only 4 support faces (highlighted with black circles) instead of 12. The margin bounds corresponding to $g_{\text{SVM}}(x;\alpha,b) \pm 1$ are also shown as dotted black lines.

It is worth noting that the size m of z^{c_k} is budgeted and intended to be much smaller than the original set of support faces $S_f^{c_k}$, e.g. $m = 2$. This is the reason why the method is called *sparse* support faces: only very few feature vectors are used to compute the solution. This is clearly beneficial for both computational cost and interpretability of the model. As a principled approximation of the original SVM discriminant function (20), it is also expected to preserve the corresponding accuracy.

The following greedy solution is adopted to solve the minimization problem (25):

1) The vector z of virtual faces is initialized with a budgeted size m;
2) β update: assuming the set of virtual faces z is constant, (25) turns out to be a regularized least squares (RLS) problem with the following solution:

$$\beta = (M + \lambda I)^{-1} + \; Ng \tag{27}$$

where

$$\underbrace{M}_{m\times m} = \frac{1}{n} S_{xz}^{\top} U S_{xz} \tag{28}$$

$$\underbrace{N}_{m\times n} = \frac{1}{n} S_{xz}^{\top} U \tag{29}$$

3) z_j update: assuming the vector of coefficients β is constant, each virtual face can be updated following the opposite direction of the gradient (numerator layout is used, as the explicit sizes suggest in the following expressions):

$$\underbrace{\frac{\partial \Omega}{\partial z_j}}_{1\times d} = \frac{1}{n} \underbrace{(h-g)^{\top}}_{1\times n} U \left(\beta_j \underbrace{\frac{\partial S_{xz_{:,j}}}{\partial z_j}}_{n\times d} + S_{xz} \underbrace{\frac{\partial \beta}{\partial z_j}}_{m\times d} \right) + \lambda \beta^{\top} \frac{\partial \beta}{\partial z_j} \tag{30}$$

with

$$\frac{\partial S_{xz}(i,j)}{\partial z_j} = 2\gamma \, S_{xz}(i,j) \underbrace{(x_i - z_j)^{\top}}_{1\times d} \tag{31}$$

$$\frac{\partial \beta}{\partial z_j} = (\underbrace{N_{j,:}}_{1\times n} g) \underbrace{\frac{\partial M_{:,j}^{-1}}{\partial z_j}}_{m\times d} + \underbrace{M_{:,j}^{-1}}_{m\times 1} g^{\top} \underbrace{\frac{\partial N_{j,:}}{\partial z_j}}_{n\times d} \tag{32}$$

$$\frac{\partial M(i,j)}{\partial z_j} = 2\gamma \, M(i,j)(x_i - z_j)^{\top} \tag{33}$$

$$\frac{\partial N(i,j)}{\partial z_j} = \begin{cases} 2\gamma \, N(i,j)(x_i - z_j)^{\top} & \text{if} \quad i = j \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

A pseudo-code of sparse SFM is presented in Algorithm 1, with the above formulas implemented in order in the functions *update_beta* and *compute_gradient*.

To sum up, the key ingredients of the algorithm are the following:

- *Dimensionality reduction*: extracting a few meaningful features out of the original face images without loss of information is arguably the most crucial step for the entire pipeline;
- *Matching score*: the function used to compare features has to be accurately chosen since the performances of the classifiers entirely rely on its efficiency.
- *Budgeted number of support faces*: since collecting a large training set is often difficult, only a small number of face templates can be exploited in practice for each identity.

IV. Experiments

A. Description of real scenes

1) Experiments performed: In order to analyze the features and the performance of the proposed method, we decided to use the COCO image captioning dataset, which contains 82,783, 40,504 and 40,775 images for training, validation and

Algorithm 1: Sparse Support Face Machine (SSFM)

Result: Learn several binary classifiers able to verify a claimed identity from face images

Input: A dataset of images $V = (v_i)_{i=1}^{n}$with the corresponding identities $Y = (c_i)_{i=1}^{n}$, a set of target identities $\mathcal{C} = (c_k)_{k=1}^{p}$, a budgeted number m of virtual support faces to be used for classification, the regularization parameters C, λ, a learning rate η, a classification threshold for each identity θ^{c_k}, a convergence condition

Output: A set of coefficients β^{c_k} and a set of support faces z^{c_k} for each identity $c_k \in \mathcal{C}$

$X, \gamma \leftarrow$ extract_features(V)
foreach *identity* $c_k \in \mathcal{C}$ **do**
  $Y^{c_k}, U \leftarrow$ binary_labels_weights(Y, c_k)
  SVM $\leftarrow$ train$(X, Y^{c_k}, \gamma, C, U)$
  $g \leftarrow$ SVM.classify(X)
  $z \leftarrow$ initialize_support_faces(X, Y^{c_k}, m)
  $S_{xz} \leftarrow$ compute_similarity(X, z, γ)
  $\beta, M, N \leftarrow$ update_beta(S_{xz}, U, λ, g)
  $h \leftarrow S_{xz}\,\beta$
  $q \leftarrow 0$
  repeat
    $j \leftarrow q \mod m$
    (1) z_j *update*
    $\frac{\partial \Omega}{\partial z_j} \leftarrow$ compute_gradient$(j, S_{xz}, M, N, h, g, U, \beta, \gamma)$
    $z_j \leftarrow z_j - \eta \cdot \frac{\partial \Omega}{\partial z_j}$
    (2) β *update*
    $S_{xz} \leftarrow$ compute_similarity(X, z, γ)
    $\beta, M, N \leftarrow$ update_beta(S_{xz}, U, λ, g)
    $h \leftarrow S_{xz}\,\beta$
    $q \leftarrow q + 1$
  until *convergence*;
  $\beta^{c_k}, z^{c_k} \leftarrow \beta, z$
end

test respectively. The images of the COCO dataset contain multiple objects in the context of complex scenes, making it a standard benchmark for assessing the performance of image captioning models. Each image in the COCO dataset has 5 human annotated captions. In order to build the vocabulary used by the extended LSTM in order to create word embeddings, we adopted the same strategy used in the reference paper for the proposed method: we built a vocabulary of words that occur at least 5 times in the COCO training set, leaving us with 10.120 unique words. The training set is originated from the following split, known as Karpahty's split, from the namesake fellow researcher: we assign 5000 images for validation and test each, after joining together the COCO original validation and training datasets, and when hold the rest out for training. The training procedure has been executed on a single machine of a HTCondor cluster whose GPU is a NVIDIA Tesla V100-PCIE-32GB. The whole training procedure took us two days to perform 35 epochs, differently from the what the authors of the paper obtained, which is 30 hours on a single NVIDIA Titan X GPU. The best model was the one corresponding to epoch 27, because the program terminated the training procedure with early stopping, in fact the validation score was not improving. Therefore, in this section, we will consider the data associated to the model obtained at epoch 25, because the improvement from 25 to 27 was neglegible. Instead of using the CiDEr metric as a way to compute validation scores like Lu et al did, we decided to employ SPICE instead, because this metric has been recently proposed for the specific task of image captioning, and captures human judgments over model-generated captions better than other automatic metrics. Moreover, for each epoch, we also computed and saved the validation score for each of the other metrics, as long as the number of epochs performed without improvement on the SPICE metric. The results for all the metrics reported in the results were computed by using COCO's captioning API [44]. We also kept track, in each set of generated captions produced at epoch n, of the visual grounding probability $1-\beta$ associated to each distinct word at each epoch and then, at the end of the training procedure, we averaged the values obtained for each distinct word. We recall that the visual grounding probability is used in order to assess whether the model learns to separate visual words in captions from non-visual words.

In order to review all the experiments that we trained the model on the cluster, please refer to the code implemented in the Jupyter notebook attached to this project, which contains the visualization of the visual grounding probabilities, as well as the spatial attention associated to each word of sampled sets of generated captions, like the one in Fig. 15.

2) Discussion of the results: In order to analyze the data that we saved during training, we decided to compare the results obtained by our implementation of the method with the ones reported in the reference paper. As a first step, we considered the visual grounding probability values associated to the sample words provided in the paper. The reference figures for this comparison are Fig. 10 and Fig. 11.

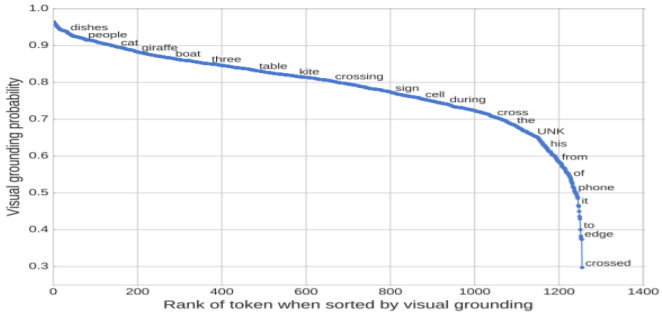

Fig. 10: Rank-probability plot reported in the reference paper for the COCO validation set.

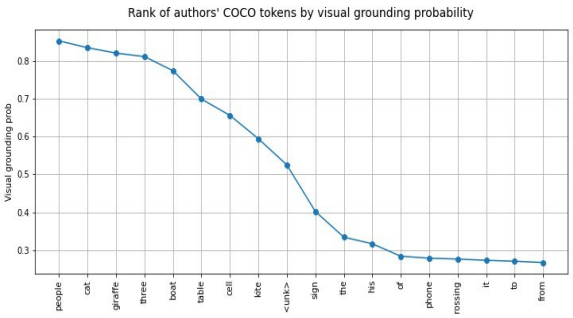

Fig. 11: Rank-probability plot of the average visual grounding probabilities computed on the validation set with our implementation when drawing the paper's cited words.

To perform the plot in Fig. 11, we looked for all the words reported in Fig. 10 in a dictionary which contained the computed average visual grounding probabilities. We found out that some of the words reported in the original paper were missing in our plot, for instance the word "dishes". We hypothesized that this happened for the following motivations:

1) We decided not to follow the standard preprocessing techniques used on the captions of the COCO dataset, which imply the usage of the NLTK tokenizer, whose documentation is reported at https://www.nltk.org/api/nltk.tokenize.html. Instead, we used the same tokenizer that was employed at evaluation time, when comparing the generated captions with the reference captions, because the reference captions needed to be tokenized again after building the vocabulary. This tokenizer is called PTB, and is contained in the COCO API that we downloaded from the previously mentioned repository.
2) The COCO dataset is maintained over time, and constantly updated, since it is a standard benchmark for various image captioning algorithms. Therefore, the semantic variance of the dataset may change over time with the corresponding generated words.

We also observed that, in general, the visual grounding probabilities obtained by us in Fig. 11 have lower values than the ones reported in Fig. 10. In particular, we can observe that the word "*sign*" has been lowered in the plot by a significant amount of visual grounding probability: from 0.8 to 0.4. This is not surely what we would expect from a state of the art model; in fact, even though the word "sign" can be efficiently predicted from the language context, on the other hand it could be for instance a street sign, and therefore, intuitively, the corresponding up-sampled attention weight vector it could acquire a lot of visual significance. We hypothesized these lower values of visual grounding probability to be produced by a combination of the following issues:

1) When we generate the next word using the LSTM described in the previous section, we use a simple greedy search to compute a score for each possible word in the vocabulary and we take the word which achieves the top score among all the others. The authors of the paper instead employ the so-called *Beam search* algorithm [35], which does not decide the predictions until we've finished decoding completely the sentence: it chooses in fact the sequence that has the highest overall score from a basket of candidate sequences. Using a simple greedy search instead of beam search may have degraded performances quite a lot (up to 10% on SPICE), and therefore may have made some words lose visual significance.
2) The previous issue may have tricked us into believing that improvements for the model would have stopped after epoch 25.

A result that we obtained which is similar to the one contained in the paper is the negative impact of word correlation on the values of the visual grounding probability. In fact, both in Fig. 10 and Fig. 11, the model cannot distinguish between words that are truly non-visual (like the word "*cell*") from the ones that are technically visual (like "*phone*") but have a high correlation with other words ("*cell-phone*") and hence chooses to not rely on the visual signal. Therefore, the model is tricked into assigning a greater visual grounding probability to a non-visual word.

Our results mimic the ones obtained in the reference paper for what concerns the classes of words associated to the visual grounding probability: in fact, the model successfully learns to attend to the image less when generating non-visual words such as "of", "his", "the", "it", whereas for visual words like "people", "cat", "giraffe", "three" and "boat", the model assigns a high visual grounding probability.

A last important observation has to be made for the "*unk*" tokens, which represents words not recognized from the vocabulary. The associated visual grounding probability is equal to 0.5 in both the plot reported in the paper and the one produced by us. This means that, on average, the attended regions which are ambiguous from the point of view of spatial significance receive a visual grounding probability value which is aligned with it.

From the plot reported in Fig. 12, we also observed that if the frequency of the word in the generated validation set increases, then the corresponding visual grounding probability decreases.

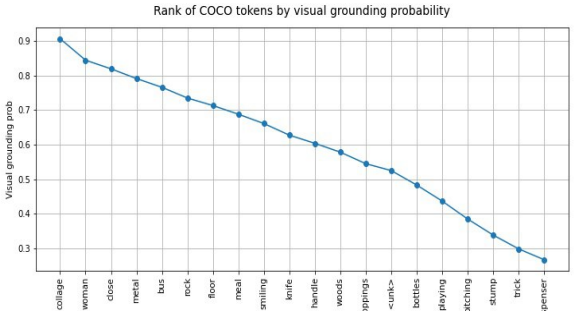

Fig. 12: Rank-probability plot of the average visual grounding probabilities on the validation set, when uniformly drawing words from the distribution sorted by ascending frequency.

In fact, as the frequency of a certain word in a generated sentence set increases, that word will be more likely to be assimilated to the class of words such as ”of”, ”the”, ”in”. We have to recall that we are trying to train a model, and therefore we are considering the most frequent words in a very wide range of possible visual context. It may therefore happen that the model struggles to attend to the correct image regions for some words.

Regarding the validation scores computed with SPICE, we noticed that as we increased the number of epochs use for training, the corresponding SPICE value computed on the validation set was increasing. In order to further improve the SPICE scores, we could implement, in the future, the beam search algorithm and run the training for a number of epochs ≥ 50, which is the value used in the reference paper. Unfortunately, we were not able to perform these two operations, due to strict time constraints of the project and to the large execution time obtained while running the training on the cluster. Another feature that we reserve for the future is implementation of *Teacher forcing* in the LSTM for the purpose of caption generation, in order to improve the convergence of the model and to generate more concise and detailed captions. In teacher forcing, if a LSTM, or a RNN in general, makes a mistake in predicting a word given the previous context, we force the generated word to be equal to the ground truth, in order to be more accurate on the whole sentence. In fact, if we get one word wrong, then the entire sentence built starting from the next timestep will be very likely to be wrong too. Training with teacher forcing could help at the early stages of training, because usually the first predictions of the model are very bad, especially with very complex models like the one introduced by Lu et al, and erros may accumulate.

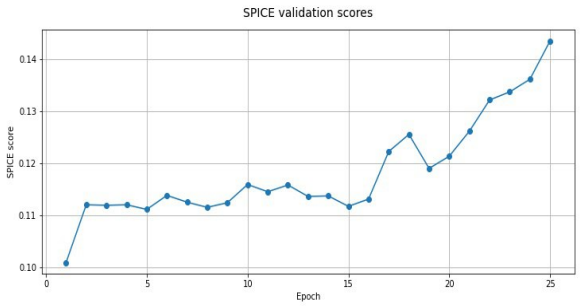

Fig. 13: Validation scores computed for the SPICE metric, when training the model for 25 epochs.

As a last interesting point on the experiments that we performed, we compared the results obtained for the most used metrics for image captioning with the ones computed by NeuralTalk 2 [45], an implementation of the Encoder-Decoder framework which follows the model from Vinyals et al. (2015). The results for both methods applied on the separate COCO test set are reported in Table I below.

The two methods perform similarly for the various metrics.

For what concerns the evaluation time on new sample images, our implementation of KWL takes 163 seconds to generate captions for all the 5000 images contained in the separate test set. Therefore, we end up with an execution time of 32 milliseconds for the generation of a single sentence for a specific image. KWL would therefore likely be able to provide high-level cognitive functions on devices with low computational power, such as the already mentioned Envision glasses or an Android device, which, despite being the original target of the project, couldn't be experimented in time.

B. Familiar faces identification

Our experiments are aimed at testing the sparse support face machine (SSFM) algorithm - implemented from scratch - on a set of familiar faces with two different feature extraction procedures:

- Principal Component Analysis (PCA) with 95% of preserved variance, as suggested in the original SSFM paper [7];
- FaceNet embedding with a Keras pretrained model on the CASIA-WebFace dataset [13] (a re-training on our machines was unfeasible considering that in the original paper the authors mentioned that the training phase took from 1,000 to 2,000 hours on a powerful CPU cluster).

Two different small datasets have been considered:

- a custom dataset of our faces - represented in fig. 14 and composed of 16 images for each of our 4 identities - to be used as 50% as training set (later referred as *train*) and the remaining 50% as test set (later referred as *test*);
- the 5 Celebrity Faces Dataset [14], composed of 93 pictures of 5 famous people, to be used as a second test set (later referred as *5cel*).

The original full frame images have been manually cropped to contain only faces and resized to 160×160 in order to match FaceNet input size. The following parameters have been used to train SSFM:

- The budget for virtual support faces was set to $m = 2$, randomly initialized from the positive class. This means that we would like to learn a discriminant function h^{c_k} based on a weighted sum of the matching scores computed between a given image and only two virtual faces;
- The parameter γ for the radial basis function was set to the number of features multiplied by the variance of the flattened training set (as suggested by scikit-learn [15]);
- The regularization parameter of the SVM was set to $C = 1$, while the class weights u^{c_k} were set to the inverse frequencies of each class (following the common practice);
- The regularization parameter for the reduction problem λ was set to 10^{-6} and the learning rate to $\eta = 0.5$, ss suggested by the paper.

[2]SPICE metrics ironically are not present, because these scores where gathered on Colaboratory and COCO's API has a problem running SPICE's JAR command on that environment.

TABLE I: Evaluation metrics on COCO's test set [2]

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr
Adaptive attention KWL	0.675	0.509	0.372	0.271	0.216	0.502	0.666
Neural Talk 2	0.747	0.590	0.453	0.345	0.260	0.555	1.058

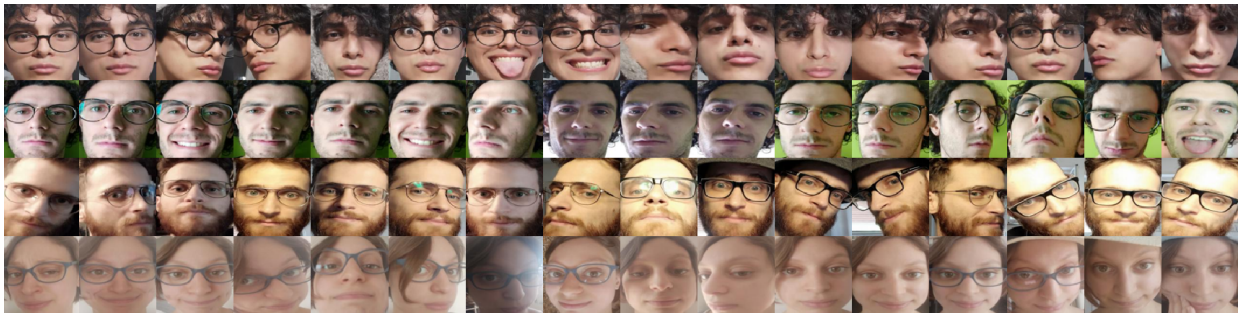

Fig. 14: The custom dataset of our faces. It is composed of 16 images of size 160×160 for each of our 4 identities.

- The convergence condition employed was the minimum relative improvement in the cost function, namely $\epsilon = 10^{-4}$ to ensure convergence in a short time.
- The user specific threshold used to get the final prediction was empirically set to a constant $\theta = 0.8$ for each identity, in order to achieve good performances either on the training set or the two test sets.

The results of the experiments are presented in Table II. Besides the canonical accuracy, sensitivity and specificity we computed also the two metrics used in face verification for each binary classification problem:

- False acceptance rate (FAR), i.e. the ratio of misclassified negative samples

$$\text{FAR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{35}$$

- False rejection rate (FRR), i.e. the ratio of misclassified positive samples

$$\text{FRR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{36}$$

where TP, FP, TN, FN stand for true positives, false positives, true negatives and false negatives, respectively. The main goal of a face verification system is to have both FAR and FRR close to zero, with a special attention on the FAR to preserve security. Moreover, we computed also a global accuracy in the multiclass setting. In order to transform SSFM's output to multiclass labels $\mathcal{C} = \{c_k\}_{k=1}^{p}$, we adopted the following decision rule:

$$\hat{y}_i = \begin{cases} \arg\max_j H_{i,j} & \text{if} \quad H_{i,j} \geq \theta \\ -1 & \text{otherwise} \end{cases} \tag{37}$$

where H is the matrix obtained by concatenation of the column vectors h^{c_k}, with $k = 1, \ldots, p$, i.e. the discriminant functions computed on the input by the single support face machines trained to identity a single identity c_k. Note that -1 is now used as the label denoting unknown identities, i.e. not included in the set $\mathcal{C}$. Such decision rule is consistent with the interpretation of the SVM learning algorithm (eq. 22) as a Tikhonov regularization for the hinge loss function, whose ideal target function leads to a Bayes decision rule of the form $sign\left[p(1|x) - p(-1|x)\right]$ [16].

Overall, the performances are good for both PCA and FaceNet feature extraction with a training time of less than 3 seconds. Different results can be noted in the multiclass accuracy where the classifier based on PCA show lower generalization capacity. This was expected since PCA is learnt on very few images taken from our custom dataset, while Facenet embedding is learnt from a huge database of faces.

V. Discussion

In this project we implemented and tested some algorithms for dense image captioning and face identification with a more ambitious mobile application in mind, intended to provide assistance to visually impaired people. The algorithms presented seem to meet our requirements of efficiently replacing some compromised visual cognitive tasks of daily utility.

A deployment of the models on Android may be considered in future projects by integrating the text-to-speech (TTS) and speech recognition (SR) modules we already developed. We would like to perform possibly real-time semantic analysis of specific scenes captured with an Android camera. For instance, the automatic real-time captioning and face identifications functionalities can trigger the TTS module to read out loud the generated image description or to signal the presence of a familiar person. These descriptions ideally should result to be fluent and should have a fine level of detail. Because the result of the two operations will be transformed into speech, the goal is to not disturb the person with continuous feedback but rather provide a detailed scene description or text reading at the person's request.

AUTHORS' CONTRIBUTIONS

F.M. and M.G. worked on dense image captioning and are the authors of par. II (Dense captioning systems) and par. IV-B (Text-to-speech module). S.B and G.L worked on face identification and are the authors of par. III (Face Identification systems) and par. IV-C (Familiar faces identification). All the students contributed equally to the other paragraphs and to the development of the respective code.

ACKNOWLEDGMENT

The students acknowledge Professor Francesca Odone who allowed them working on this project and provided useful pieces of advice from the get-go, and CERN organization for letting M.G. and F.M. train the KWL model for over 50 hours on their HTCondor cluster.

REFERENCES

[1] Envision glasses official webpage https://www.letsenvision.com/glasses, accessed on 24th June, 2020.
[2] OrCam official page https://www.orcam.com/en/myeye2/, accessed on 24th June, 2020.
[3] Facial Recognition and Biometric Technology Moratorium Act of 2020, https://www.markey.senate.gov/imo/media/doc/acial\%20Recognition\%20and\%20Biometric\%20Technology\%20Moratorium\%20Act.pdf
[4] Buolamwini, J., & Gebru, T. (2018, January). Gender shades: Intersectional accuracy disparities in commercial gender classification. In Conference on fairness, accountability and transparency (pp. 77-91).
[5] Wrongfully Accused by an Algorithm, The New York Times, https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html, accessed on July 22, 2020.
[6] *A new US bill would ban the police use of facial recognition*, MIT Technology Review, https://www.technologyreview.com/2020/06/26/1004500/a-new-us-bill-would-ban-the-police-use-of-facial-recognition/ accessed on July 22, 2020.
[7] Biggio, B., Melis, M., Fumera, G., & Roli, F. (2015, May). Sparse support faces. In 2015 International Conference on Biometrics (ICB) (pp. 208-213). IEEE.
[8] Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1701-1708).
[9] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).
[10] Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. CMU School of Computer Science, 6(2).
[11] Balaban, S. (2015, May). Deep learning and face recognition: the state of the art. In Biometric and Surveillance Technology for Human and Activity Identification XII (Vol. 9457, p. 94570B). International Society for Optics and Photonics.
[12] Murphy K., Machine learning: a probabilstic perspective, MIT Press, 2012
[13] Face Recognition using Tensorflow, https://github.com/davidsandberg/facenet
[14] 5 Celebrity Faces Dataset, https://www.kaggle.com/dansbecker/5-celebrity-faces-dataset
[15] PCA class, scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[16] Rosasco L., *Introductory machine learning notes*, http://lcsl.mit.edu/courses/ml/1819/MLNotes.pdf
[17] P. Anderson et al. 2016. Spice: Semantic propositional image caption evaluation. In Proceedings of the European Conference on Computer Vision (ECCV). 382–398
[18] Nagel H.(1998). "From image sequences towards conceptual descriptions," in Image Vis. Comput., vol. 6, no. 2, pp. (59–74), IEEE.
[19] Khan, M. & Gotoh, T. (2012). "Describing video contents," in Proc.Workshop Innovative Hybrid Approaches Process. Textual Data, 1224 2012, pp. (27–35).
[20] Y. Yang, C. L. Teo, H. Daume´ III, and Y. Aloimonos. (2011). "Corpus-guided sentence generation of natural images," in Textual Data, in Proc.Conf. EmpiricalMethods Natural Lang. Process., 2011, pp. (444–454).
[21] V. Ordonez, G. Kulkarni, & T. L. Berg. (2011). "Im2text: Describing images using 1 million captioned photographs," in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., 2011, pp. (1143–1151).
[22] R.Mason and E. Charniak, "Domain-specific image captioning," in Proc. 18th Conf. Comput. Lang. Learn., 2014, vol. 1, pp. (11–20).
[23] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded compositional semantics for finding and describing images with sentences," in Trans. Assoc. Comput. Linguistics, vol. 2, pp. (207–21)8, 1180 2014.
[24] B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu, "I2T: Image parsing to text description," Proc. IEEE, vol. 98, no. 8, pp. (1485–1508), Aug. 2010.
[25] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi, "Collective generation of natural image descriptions," in Proc. 50th Annu. Meeting Assoc. Comput. Linguistics, 2012, vol. 1, pp. (359–368).
[26] R. Kiros, R. Salakhutdinov, and R. S. Zemel, "Unifying visual-semantic embeddings with multimodal neural language models," Trans. Assoc. Comput. Linguistics, 2015.
[27] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Jun. 2015, pp. (3128–3137).
[28] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in Proc. IEEE Conf. Comput. Vision Pattern Recognit., Boston, MA, USA, Jun. 7–12, 2015, pp. (3156–3164).
[29] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, Youngjoon Yoo, "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features", 2019.
[30] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (M-RNN)," Proc. Int. Conf. Learning Representations (ICLR), 2015.
[31] X. Chen and C. L. Zitnick, "Mind's eye:Arecurrent visual representation for image caption generation," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2015, pp. (2422–2431).
[32] K. Xu et al., "Show, attend and tell: Neural image caption generation with visual attention," in Proc. 32nd Int. Conf. Mach. Learn., 2015, pp. (2048–2057).
[33] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. (4651–4659).
[34] J. Lu, C. Xiong, D. Parikh, and R. Socher. "Knowing when to look: Adaptive attention via a visual sentinel for image captioning". In CVPR, 2017.
[35] https://en.wikipedia.org/wiki/Beam_search
[36] https://nlp.stanford.edu/software/tokenizer.shtml
[37] https://github.com/jiasenlu/AdaptiveAttention
[38] https://github.com/fawazsammani/knowing-when-to-look-adaptive-attention
[39] https://github.com/yufengm/Adaptive
[40] https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning
[41] https://medium.com/swlh/image-captioning-using-attention-mechanism-f3d7fc96eb0e
[42] https://www.tensorflow.org/tutorials/text/image_captioning
[43] https://towardsdatascience.com/what-is-gradient-clipping-b8e815cdfb48
[44] https://github.com/ruotianluo/coco-caption
[45] https://github.com/karpathy/neuraltalk2

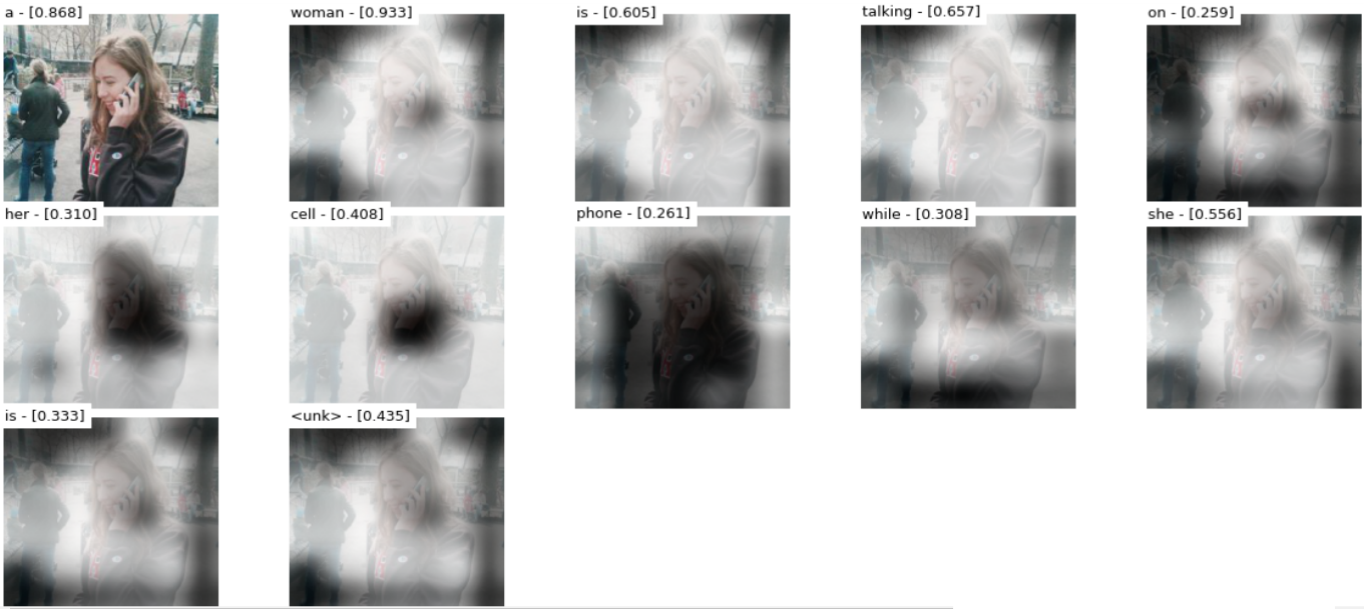

Fig. 15: Visualization of spatial attention maps associated to image regions at each word generated.

TABLE II: SSFM performances

		PCA					FaceNet				
		acc	sens	spec	FAR	FRR	acc	sens	spec	FAR	FRR
Federico	train	1.00	1.00	1.00	0.00	0.00	0.97	0.88	1.00	0.00	0.12
	test	0.97	0.88	1.00	0.00	0.12	1.00	1.00	1.00	0.00	0.00
	5cel	0.97	-	0.97	0.03	-	1.00	-	1.00	0.00	-
Gianvito	train	1.00	1.00	1.00	0.00	0.00	0.94	0.75	1.00	0.00	0.25
	test	0.94	0.75	1.00	0.00	0.25	0.94	0.75	1.00	0.00	0.25
	5cel	0.93	-	0.93	0.07	-	1.00	-	1.00	0.00	-
Matteo	train	0.97	0.88	1.00	0.00	0.12	0.97	0.88	1.00	0.00	0.12
	test	0.94	0.75	1.00	0.00	0.25	0.94	0.75	1.00	0.00	0.25
	5cel	0.86	-	0.86	0.14	-	0.99	-	0.99	0.01	-
Sofia	train	0.97	0.88	1.00	0.00	0.12	0.97	0.88	1.00	0.00	0.12
	test	0.97	0.88	1.00	0.00	0.12	0.94	0.75	1.00	0.00	0.25
	5cel	0.97	-	0.97	0.03	-	0.94	-	0.94	0.06	-
Mean value	train	0.98	0.94	1.00	0.00	0.06	0.96	0.85	1.00	0.00	0.15
	test	0.95	0.82	1.00	0.00	0.18	0.95	0.82	1.00	0.00	0.18
	5cel	0.93	-	0.93	0.07	-	0.98	-	0.98	0.02	-
		Multiclass accuracy									
	train	0.94					0.84				
	test	0.81					0.81				
	5cel	0.72					0.92				