# Deep Learning Project: Neural Image Captioning

**Ryan Boustany**
ENSAE Paris
ryan.boustany@ensae.fr

**Emma Sarfati**
ENSAE Paris
emma.sarfati@ensae.fr

## Abstract

Recent amazing progress has been made in the field of computer vision and natural language processing. Deep learning has come to be the common thread of these improvements. In this paper, we propose to handle a task that gathers both disciplines: generating automatically the caption of an image, by using convolutional and recurrent neural networks. CNN are usually used for applications in image processing and computer vision, due to their well-adapted structure for 2D and 3D tensors, such as images. RNN are usually used for NLP tasks, such as language modelling, as their recurrent structures fit well to text sequences. Image Captioning aims at using images for predicting text.

## Introduction

Image Captioning has become a more and more popular task and research topic for Object Recognition and other Vision-Language Task. The current state-of-the-art is pretty affluent and leads to impressive result. Microsoft research team has notably published the OSCAR model [6] which proposes a whole new type of training process that leads to the most accurate results, which uses object tags detected in images as anchor points to significantly ease the learning of alignments. However, usually, the vanilla method to achieve image captioning relies on learning the best features of the image thanks to a convolutional neural network[1], and then using these features in a recurrent neural network[2] structure to learn the target caption. In most of the cases, an attention mechanism is used to learn attention weights on specific parts of the images to detect the right word to use ([10], [1]).
In our project, we aim at building our own structure, inspired by the original architecture proposed in [10], in a simplified way. In a first notebook, we design our architecture with our own training and evaluation procedures. In a second notebook, we work directly on the model of Xu et al., but the codes for this second model are not totally our work.

## 1 Framework

### 1.1 Problem

Our objective is to generate automatically the caption, given an image. That is, from a practical point of view, we aim at generating a vector of sequence $(x_1, ..., x_T)$ given a 3D tensor $z = (z_1, z_2, z_3) \in \mathbb{R}^{n \times n \times 3}$. To simplify matters, we will consider that the input tensor is squared. In a nutshell, our problematic is a **text generation problem**, using **images features**. Generally, with the objective of generating textual sequences, deep learning comes to help. Recurrent neural networks structures are feedforward neural networks which take into account a temporal (or sequential) dimension during the training. For our training, the input data that are available are the tensor image $z = (z_1, z_2, z_2)$ and the target caption on which we will base the learning process. We denote this caption as a ground-truth $x = (x_1, ...x_T)$. Our goal is to find a neural network-based structure that is able to take

[1] first published by Fukushima in 1980 [3]
[2] first introduced in [8]

into account the image and the caption features to generate a new caption during training. The original paper proposed by Xu et al. [10] depicts an intuitive architecture, based on an encoder-decoder that takes the convolved image as output of the encoder (so the initialized hidden state of the decoder), and an LSTM architecture as a decoder.

### 1.2 Dataset

We worked on the COCO Dataset [7]. The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. We use the 2017 version of COCO, which contains a train/validation split of 118K/5K annotated images.

### 1.3 Captions input

Our captions inputs are encoded as vectors of tokenized words, which are all of size 51 with a zero-padding at the end of each vector. Our vocabulary size is 36,303. Our objective is to generate a whole caption. At this aim, we consider a **conditional language generation** model, which has to predict, at each timestep $t$, the probability $\mathbb{P}(y_t|h_{t-1}, y_{t-1})$ where $y_t$ is the $t^{th}$ true word in the caption and $h_t$ is the hidden state of the recurrent structure that is used. The latter is illustrated in the following schema.

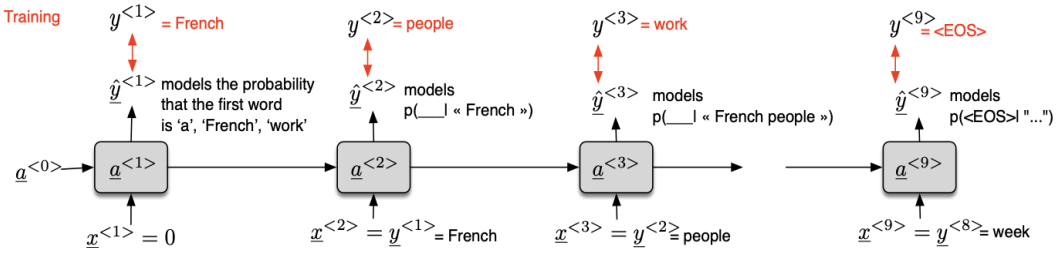


Figure 1: Recurrent neural network structure used for our conditional language model.

Formally, for just one example of the dataset and following the notations of the previous figure, our RNN is defined $\forall t \in \{1, ..., T\}$ as:

$$\begin{aligned}
\underline{x} &= \{\underline{x}^{<1>}, \underline{x}^{<2>}, \dots \underline{x}^{<T>}\} \text{ the input sequence} \\
\underline{a}^{<0>} &= \text{initialized vector of shape } (n_a, 1) \\
\underline{a}^{<t>} &= f_a(W_{ax}\underline{x}^{<t>} + W_{aa}\underline{a}^{<t-1>} + b_a) \\
\underline{\hat{y}}^{<t>} &= f_y(W_{ya}\underline{a}^{<t>} + b_y)
\end{aligned}$$

The idea is that the sequential dimension of the inputs is taken into account as the hidden state of the neural network, $\underline{a}^{<t>}$, is defined by the word at time $t$, but also by the previous words, illustrated by $\underline{a}^{<t-1>}$. The matrix $W_{aa}$ is the link between a word and all its previous ones. Matrix $W_{ax}$ maps the input $\underline{x}^{<t>}$ to the space of hidden states $\underline{a}^{<t>}$, and $W_{ya}$ maps the hidden states to the final input $\underline{\hat{y}}^{<t>}$. The weights and biases are updated using backpropagation through time (BPTT) [9], which explicit formulas can be computed using a formula of total derivative as the usual chain rule cannot be used due to the temporal dependencies between hidden states.
Now that we have a language model that can be trained using the input captions, the idea is to link the input images to this language generation model. Indeed, our goal is to have a model that takes the **sole image as an input**. The next section deals with the encoding and using of the image input and how to link it with the caption.

### 1.4 Images input

In the COCO Dataset, images have different sizes. We decided to reshape them to have all equal size of (256,256,3). The idea is to retrieve the most relevant features related to the input images. A well-known way to do it in deep learning is to use Convolutional Neural Networks. CNN allow

to retrieve a set of relevant features that come from the original image by using **convolutions** (and max pooling in practice). For computational reasons and also better features extraction, we will use a pre-trained CNN, precisely the ResNet-101 [4], which contains 347 layers of convolutions, max pooling batch normalization and residual connections, which are found to greatly improve gradient flow. The network has 44,654,504 parameters.

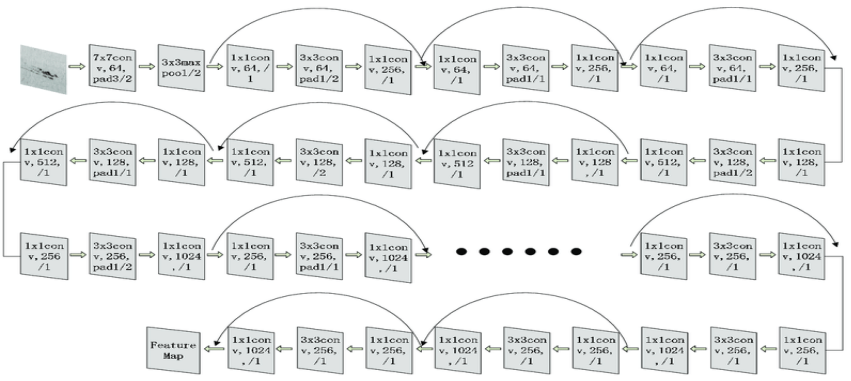


Figure 2: ResNet-101 architecture.

## 2 Models

### 2.1 Baseline model

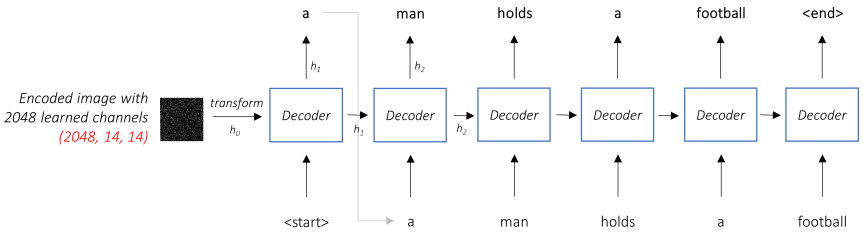


Figure 3: Our baseline structure. The convolved image is used as the initial hidden state of our RNN Decoder.

Our objective is to build a neural structure that is able to predict the probability of the next word at each timestep $t$. However, with long captions for instance, the phenomenon of vanishing gradient can happen during the Backpropagation through time. Retaking the notations introduced in the first part, we recall that the total loss formula is given by

$$\mathcal{L} = \sum_t \mathcal{L}^{\langle t \rangle} \left( y^{\langle t \rangle}, \hat{y}^{\langle t \rangle} \right)$$

The derivative of the total loss with respect to the shared matrix $W_{aa}$ of each hidden state $a^{<t>}$

$$\frac{\partial \mathcal{L}}{\partial W_{aa}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial W_{aa}}$$

And using the formula of total derivative, after computations, we obtain that

$$\frac{\partial \mathcal{L}^{\langle t \rangle}}{\partial W_{aa}} \propto \sum_{k=0}^{t} \left( \prod_{i=k+1}^{t} \frac{\partial a^{\langle i \rangle}}{\partial a^{\langle i-1 \rangle}} \right) \frac{\partial a^{\langle k \rangle}}{\partial W_{aa}}$$

which means that at each timestep $t$, the derivative of the loss with respect to $W_{aa}$ implies a product of the derivative of each hidden state at $a_i$ with respect to its previous hidden state $a_{i-1}$. If $\left| \frac{\partial a^{\langle i \rangle}}{\partial a^{\langle i-1 \rangle}} \right| < 1$

then the product goes to 0 exponentially fast. If $\left|\frac{\partial a^{\langle i \rangle}}{\partial a^{\langle i-1 \rangle}}\right| > 1$ then the product goes to infinity exponentially fast. With long sentences, the problem can be all the more stressed, as the product becomes huge or small very fast.
A famous type of cell to handle this vanishing/exploding gradient problem is the Gated Recurrent Unit [8], which is a modern version of the Long-Short Term Memory cell [5], with less parameters.

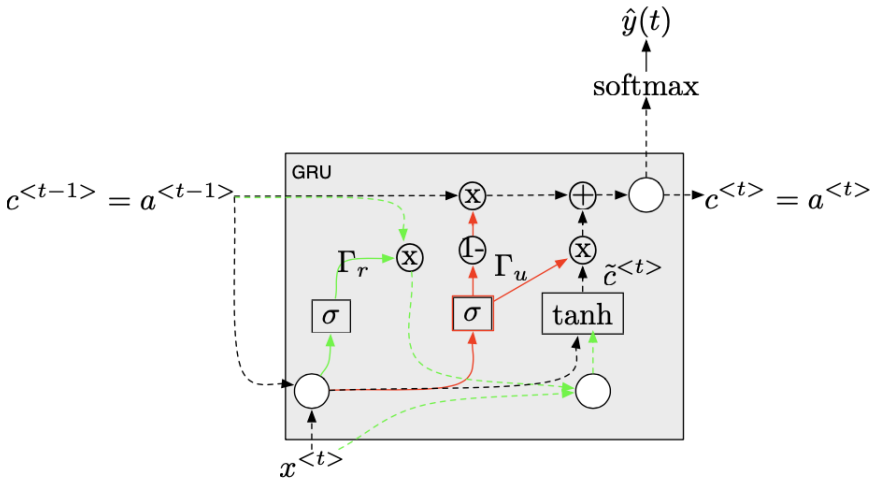


Figure 4: The GRU cell.

The structure is the following:
Relevant gate

$$\Gamma_r = \sigma\left(W_r\left[c^{<t-1>}, x^{<t>}\right] + b_c\right)$$

Candidate cell value

$$\tilde{c}^{<t>} = \tanh\left(W_c\left[\Gamma_r \odot c^{<t-1>}, x^{<t>}\right] + b_c\right)$$

Update gate

$$\Gamma_u = \sigma\left(W_c\left[c^{<t-1>}, x^{<t>}\right] + b_c\right)$$

Cell update

$$c^{<t>} = \Gamma_u \odot \tilde{c}^{<t>} + (1 - \Gamma_u) \odot c^{<t-1>}$$

Output

$$c^{<t>} = a^{<t>}$$

With this cell, we obtain that $\frac{\partial c^{\langle t \rangle}}{\partial c^{\langle t-1 \rangle}} = \text{diag}\left(1 - \Gamma_u\right)$, which prevents from vanishing or exploding gradient as $\Gamma_u$ takes values between 0 and 1. We choose the GRU cell in our implementation for decoding each word at each timestep $t$. The only difference with a classical RNN is that we initialize our network, $a^{<0>}$, with the projection of the convolved image on the space of a dimension equal to the chosen decoder dimension (called decoder_dim)[3]. The output of the GRU cell is the next hidden state of the decoder, which has a shape of (batch_size, decoder_dim). We then decode all the sentence in this way, without the risk of vanishing or exploding gradient. The results of this first model is available in the last section of this report.

### 2.2 Model with visual attention

Instead of only considering the initial hidden state as a feature from the image input, the idea introduced in the paper of Xu et al. [10] is to consider a weighted version of the input image at each timestep. This idea is adapted from neural machine translation models that use an attention mechanism to generate text translations [1] by weighting the words of the original sentence. Using the notations of the figure below, we replace $h_0 = \text{Encoder(image)} \in \mathbb{R}^{batch\_size \times 14 \times 14 \times 2048}$ by a

[3] all these implementations details are provided in our notebooks.

local version $v^{\langle\tau\rangle} = \sum_t \alpha^{\langle\tau,t\rangle} h_{0,t}$ with $\tau$ being a timestep at the decoding process, $t$ being a position on a 14×14 square, and:

$$\alpha^{\langle\tau,t\rangle} = \frac{\exp\left(e^{\langle\tau,t\rangle}\right)}{\sum_{t'=1}^{14\times14} \exp\left(e^{\langle\tau,t'\rangle}\right)} \quad \text{with} \sum_t \alpha^{\langle\tau,t\rangle} = 1$$

with $e^{\langle\tau,t\rangle}$ being computed using a feedforward neural network. Note that we index the timestep at the top for caption features ($v^{\langle\tau\rangle}$), and at the bottom for image feature ($h_{0,t}$). The attention weights $\alpha^{\langle\tau,t\rangle}$ can be interpretated as follows: when generating information at time $\tau$, how much do we have to pay attention on information present in pixel $t$ of the encoder's output, with $t \in \{1, ..., 14 \times 14\}$. In fact, the dimension of $h_0$ is $14 \times 14 \times 2048$, not $14 \times 14$, so we'd rather talk about a "pixel position" on a $14 \times 14$ grid instead of the pixel itself (due to the depth of 2048). In the original use of this model for machine translation, the attention must be payed to an original sentence, so $t$ represents each timestep of the sentence we want to translate. More mathematical details can be found in the original paper [10], but the main difficulty lies in the implementation part and not the theoretical one, notably with when it comes to deal with dimensions. In the following we detail the decoding process. Note that encoder_dim=2048, num_pixels=14×14, attention_dim=512.
For each decoding timestep $\tau$:

- We pass the encoder's output and the previous hidden state $a^{\langle\tau-1\rangle}$ through an Attention Layer. We recall that the flattened encoder output has shape (batch_size, 14×14, 2048), and decoder's previous hidden state has shape (batch_size, decoder_dim). These elements are both projected onto a space of dimension attention_dim thanks to a fully connected. It gives us features of respective shapes (batch_size, 14×14, attention_dim) and (batch_size, attention_dim). These features are additioned, passed through a feedforward+softmax with a single output, which leads to an output shape of (batch_size, 14×14) for the attention weights. We also apply these attention weights to the encoder's output by multiplying them and summing on num_pixels dimension, which leads to another output of shape (batch_size, encoder_dim). The outputs of the attention layer have shapes (batch_size, 14×14) for the $\alpha^{\langle\tau,t\rangle}$ and (batch_size, encoder_dim) for the encoder's output with applied weights.
- We apply an LSTM cell which has the following inputs:
  - Input at time $\tau$: concatenated attention weighted encoded image+word at time $\tau$ (**see figure 5**).
  - Previous hidden and cell states $a^{\langle\tau-1\rangle}$ and $c^{\langle\tau-1\rangle}$.

  we then retrieve the next hidden state $a^{\langle\tau\rangle}$ and cell state $c^{\langle\tau\rangle}$.
- We store the word prediction by applying a fully connected + dropout of output shape vocab_size to the hidden state.
- We repeat the process until maximum caption length.

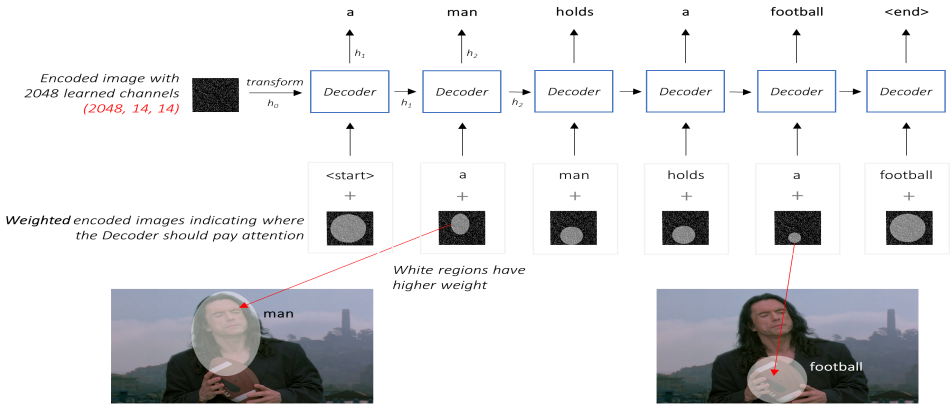


Figure 5: The decoder with visual attention weights [10].

## 3 Results

The inference part is realized by initializing the RNN with the input image and decoding word after word using the trained weights of the GRU (or the LSTM+attention for the second model). The predicted word is given by passing each computed hidden state of the decoder through a fully connected layer with output shape equal to the vocabulary, and taking the maximal index of it. In our notebooks, we use a particular algorithm to decode with more confidence: the Beam Search algorithm [2]. We ran our experiments on a MacBook Pro with 8 Go of RAM and, 1.4 GHz Intel Core i5 4-cores processor.

### 3.1 First model

The hyperparameters were chosen intuitively or regarding the common settings in deep learning. The decoder learning rate was set to 0.0004, the number of epochs was set to 30 and the embedding dimension was fixed at 512, the same for the decoder dimension. It was possible to increase these sizes, which could have led to better results. But this model is a baseline and we wanted to compare it wisely with the attention one, so we used the same hyperparameters. The results ar every variable.

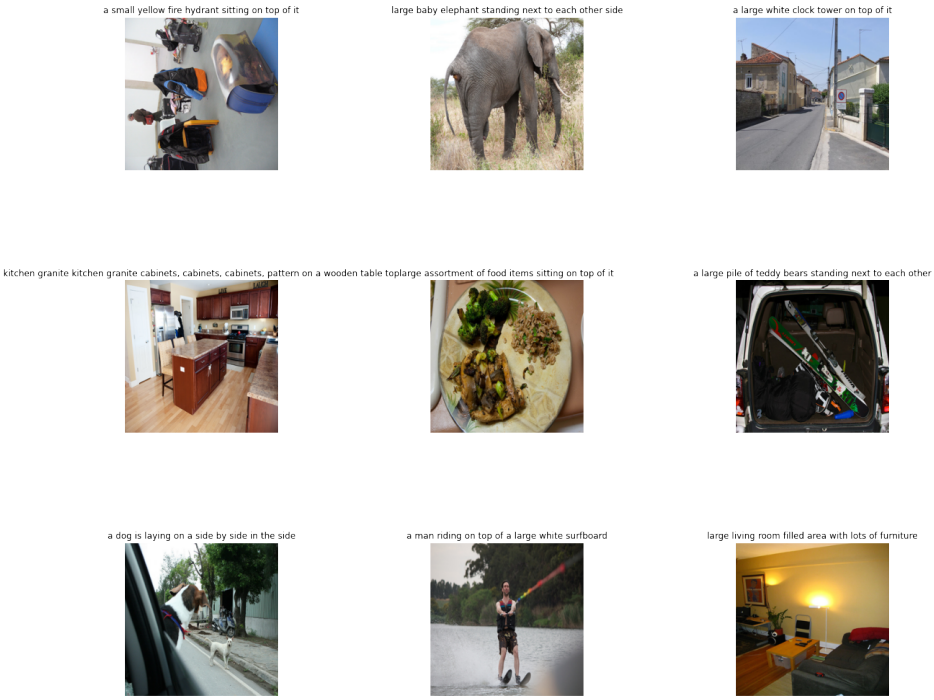


Figure 6: Results of the GRU-based model.

Note that we also tested our model on other samples of the COCO dataset during our experiments, which led to the same type of results. A caption can be very clear and relevant for one image and totally out of context for another one. A general observation is that this model is able to capture the most important elements of the images; but not always able to build a grammatically-correct formulation of it. It also happens that the caption is beside the point (see the sixth example in 6).

### 3.2 Second model

The hyperparameters were chosen according to the first model. The decoder learning rate was set to 0.0004, the number of epochs was set to 30 (in the notebook we let the version with 40 epochs which gives almost the same accuracy), and the embedding dimension was fixed at 512, the same for the decoder dimension and the attention layer dimension. As we use gradient clipping to prevent exploding gradient, we chose a clipping parameter of 5.

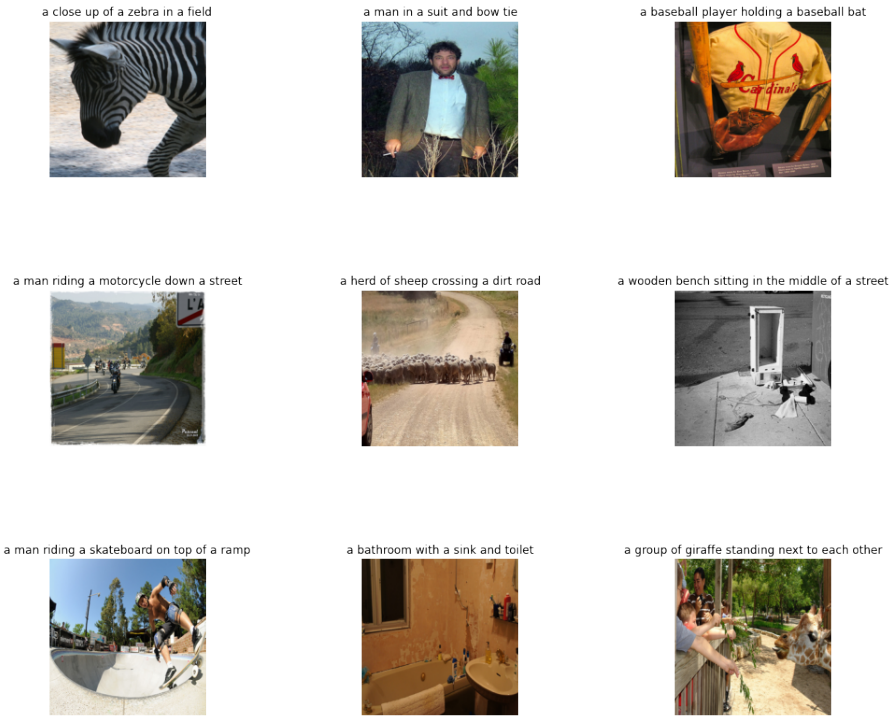


Figure 7: Results of the LSTM with visual attention model.

We can clearly note a huge difference with our baseline model. All the captions seem coherent, and at least always linked with the true context of he image. It still makes some mistake (last picture, confusion between a group of giraffes and a group of people looking at giraffes), but seems to be able to map properly an image to a caption. To test the ability of our model to generalize, we test with two of this report's authors Facebook pictures.


Figure 8: Predicted captions on Facebook pictures.

The results are surprisingly very accurate. The model took less than 30 minutes to train on a Google Colab GPU, on a training database of 6666 examples, and seems to achieve great results. However, we can denote a mistake with the repetition of the hat element on Ryan's picture.

## Conclusion

We studied the image captioning state-of-the-art and focused on the commmon ways to handle the task, through different recent and older research papers. We tried to propose a first model based on the simpler way to achieve it, and a second model based on a more elaborated approach, following [10]. Our baseline model gave encouraging results, while the second gives pretty impressive ones. We think that with more parameters tuning and higher number of epochs, greater results could have been reached, be it for the first or the second model. This work allowed us to review the whole bunch of the deep learning landscape: feedforward neural networks, convolutional neural networks, and recurrent neural networks.

## References


[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[2] Markus Freitag and Yaser Al-Onaizan. "Beam search strategies for neural machine translation". In: *arXiv preprint arXiv:1702.01806* (2017).

[3] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[4] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[5] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[6] Xiujun Li et al. "Oscar: Object-semantics aligned pre-training for vision-language tasks". In: *European Conference on Computer Vision*. Springer. 2020, pp. 121–137.

[7] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X.

[9] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[10] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International conference on machine learning*. PMLR. 2015, pp. 2048–2057.