Deep Learning 2

Image Captioning

Our goal: produce a caption from an image

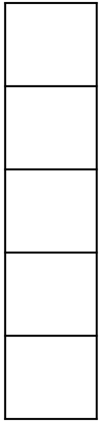
MODEL →

"a group of children run a footrace in the snow"

Overall architecture of our model

Encoder

Truncated Resnet101 (last 2 layers are removed)



Decoder

LSTM + Attention + Beam search

[<start>, 'a', 'group', 'of', 'children', 'run', 'a', 'footrace', 'in', 'the', 'snow', <end>, <pad>, <pad> ...]

Object: image (jpg, png)

Object: feature map (encoded version of our image)

Object: sequence of strings

Remark: starts with <start> and ends with <end>, + <pad>

Architecture based on Xu, Kelvin, et al. "Show, attend and tell [...]", implementation based on this repo

Focus on the encoding

Encoder

Truncated Resnet101 (last 2 layers are removed)

Decoder

LSTM + Attention + Beam search

[<start>, ‘a’, ‘group’, ‘of’, ‘children’, ‘run’, ‘a’, ‘footrace’, ‘in’, ‘the’, ‘snow’, <end>, <pad>, <pad> …]

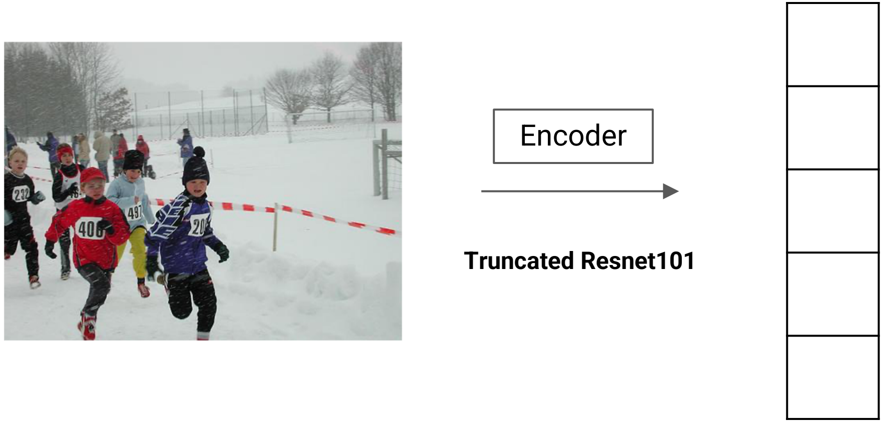
Object: image (jpg, png)

Object: feature map (encoded version of our image)

Object: sequence of strings

Remark: starts with <start> and ends with <end>, + <pad>

Architecture of the encoder



> We used **transfer learning** as our backbone model is a pretrained **Resnet101 (trained on ImageNet)**

> **The last two layers** (softmax and dense) **are removed**: enables to **extract features** from images

> This model takes as **input** a float tensor of size (batch size, 3, 256, 256)

> The **output** is a tensor of size (batch size, 14, 14, 2048)

> We could **replace Resnet101** by other pretrained models, **or fine-tune** this part of the model for our specific needs

Focus on the decoder

Encoder

Truncated Resnet101 (last 2 layers are removed)

Decoder

LSTM + Attention + Beam search

[<start>, ‘a’, ‘group’, ‘of’, ‘children’, ‘run’, ‘a’, ‘footrace’, ‘in’, ‘the’, ‘snow’, <end>, <pad>, <pad> …]

Object: image (jpg, png)

Size: can be resized and interpreted as an float tensor of size (1, 3, 256, 256)

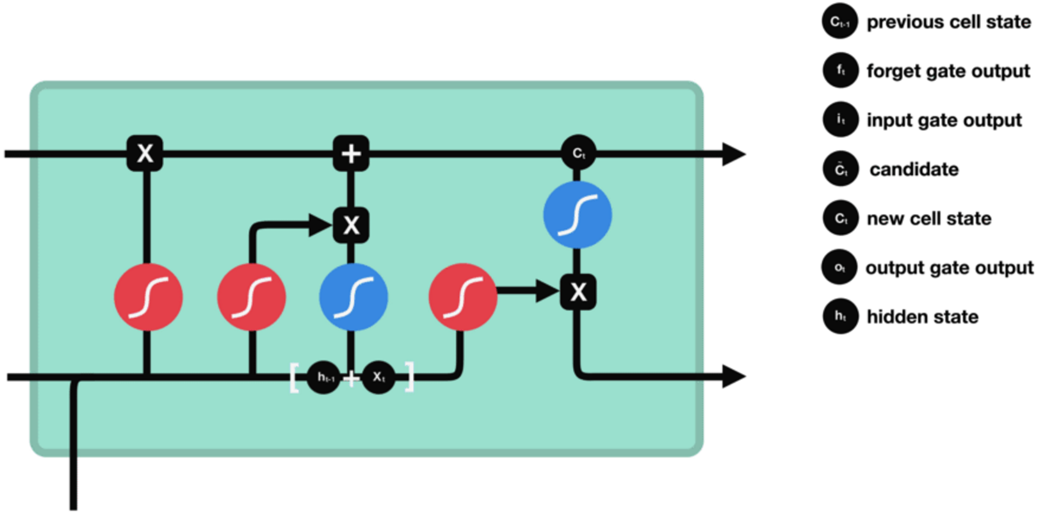
Object: feature map (encoded version of our image)

Size: (1, 14, 14, 2048)

Object: sequence of strings

Remark: starts with <start> and ends with <end>

Overall architecture of the decoder: LSTM

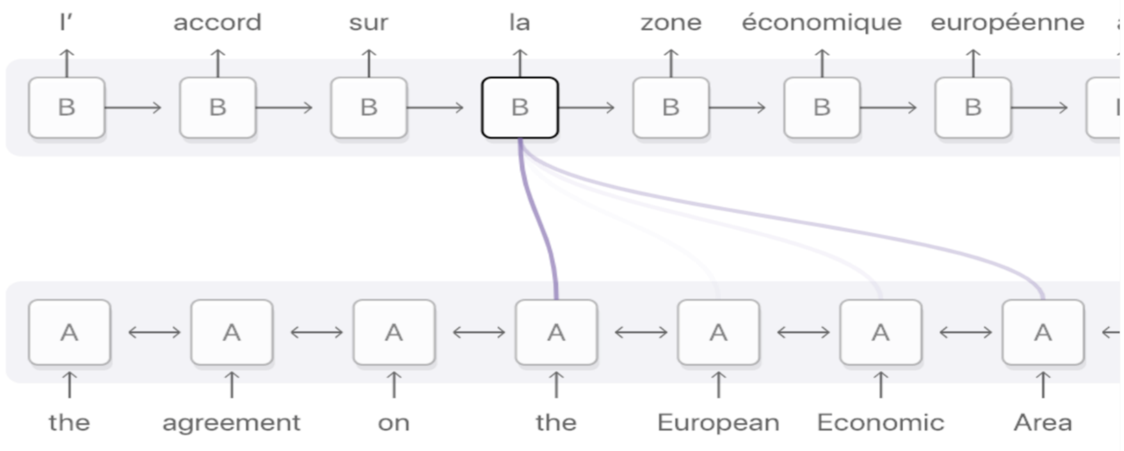


> In the context of RNN, problems of **vanishing or exploding gradients** may lead to a difficult learning

> **Long Short Term Memory (LSTM)** networks may avoir this problem by preserving useful information and getting rid of useless ones thanks to gates

> Useful/Useless information are selected thanks to weights that are learnt during the training thanks to **backpropagation**

Focus on the Attention mechanism



Alignment for the French word 'la' is distributed across the input sequence but mainly on these 4 words: 'the', 'European', 'Economic' and 'Area'. Darker purple indicates better attention scores

> **Attention** is an interface between encoder and decoder that provides decoder with information from every encoder hidden state

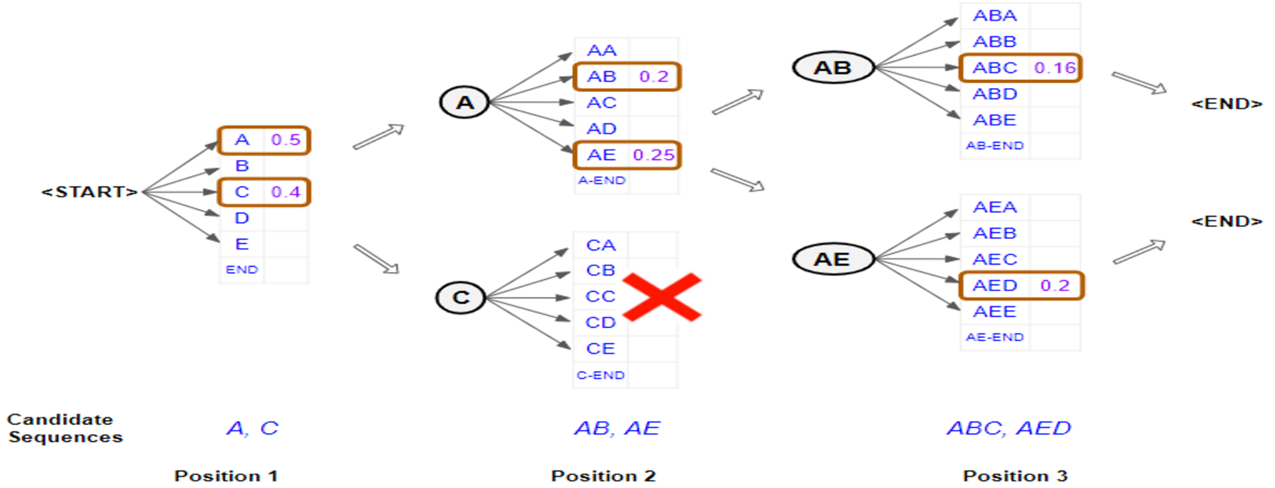
> With this setting, the model is then able to focus on a specific part of the input where the information is concentrated

> **Attention weights** are calculated for each hidden state showing the influence of the encoder hidden state on next word

> Multiple applications such as **Image Captioning**, **Video description** and **Speech Recognition**

Source: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

Focus on the beam search



> **Beam Search** is an algorithm that makes it possible to predict the next character/word in a sequence to sequence model

> Unlike **Greedy Search** that takes at each iteration the word/character with highest probability, Beam Search iteratively takes the N possibilities and computes tree probabilities for each possibility at each time, finally returning the highest one

> This technique makes Beam Search slower, but more accurate than Greedy Search as it takes into account all possibilities

> Has many applications above **Image Captioning** such as **Speech-to-text** or **Translation**

Source: https://towardsdatascience.com/foundations-of-nlp-explained-visually-beam-search-how-it-works-1586b9849a24

Possible improvements/related work

> Use **other pretrained CNNs** instead of Resnet101 to see how/if it affects the performance

> **Fine-tune the backbone model** to see how/if it affects the performance

> **Play with the beam size** (size of the tree when we explore possible sentences) to see how/if it affects the performance

> Adapt this model to **video description/captioning**

> **Explore generative models to produce images from captions/short descriptions**

Demo

> See Streamlit app