In the following, we present additional material about our $M^2$ Transformer model. In particular, we provide additional training and implementation details, further experimental results, and visualizations.

1. Additional implementation details

Decoding optimization. As mentioned in Sec. 3.3, during the decoding stage computation cannot be parallelized over time as the input sequence is iteratively built. A naive approach would be to feed the model at each iteration with the previous $t-1$ generated words, $\{w_0, w_1, \ldots, w_{t-1}\}$ and sample the next predicted word $w_t$ after computing the results of each attention and feed-forward layer over all timesteps. This in practice requires to re-compute the same queries, keys, values and attentive states multiple times, with intermediate results depending on $w_t$ being recomputed $T-t$ times, where $T$ is the length of the sampled sequence (in our experiments $T$ is equal to 20).

In our implementation, we revert to a more computationally friendly approach in which we re-use intermediate results computed at previous timesteps. Each attentive layer of the decoder internally stores previously computed keys and values. At each timestep of the decoding, the model is fed only with $w_{t-1}$, and we only compute queries, keys and values depending on $w_{t-1}$.

In PyTorch, this can be implemented by exploiting the register_buffer method of nn.Module, and creating buffers to hold previously computed results. When running on a NVIDIA 2080Ti GPU, we found this to reduce training and inference times by approximately a factor of 3.

Vocabulary and tokenization. We convert all captions to lowercase, remove punctuation characters and tokenize using the spaCy NLP toolkit[^1]. To build vocabularies, we remove all words which appear less than 5 times in training and validation splits. For each image, we use a maximum number of region feature vectors equal to 50.

Model dimensionality and weight initialization. Using 8 attentive heads, the size of queries, keys and values in each head is set to $d/8 = 64$. Weights of attentive layers are initialized from the uniform distribution proposed by Glorot et al. [3], while weights of feed-forward layers are initialized using [4]. All biases are initialized to 0. Memory vectors for keys and values are initialized from a normal distribution with zero mean and, respectively, $1/d_k$ and $1/m$ variance, where $d_k$ is the dimensionality of keys and $m$ is the number of memory vectors.

2. Additional experimental results

Memory vectors. In Table 1, we report the performance of our approach when using a varying number of memory vectors. As it can be seen, the best result in terms of BLEU, METEOR, ROUGE and CIDEr is obtained with 40 memory vectors, while 80 memory vectors provide a slightly superior result in terms of SPICE. Therefore, all experiments in the main paper are carried out with 40 memory vectors.

Encoder and decoder layers. To complement the analysis presented in Sec. 4.3, we also investigate the performance of the $M^2$ Transformer when changing the number of encoding and decoding layers. Table 2 shows that the best

[^1]: https://spacy.io/
Figure 1: Sample nocaps images and corresponding predicted captions generated by our model and the original Transformer. For each image, we report the Open Images object classes predicted by the object detector and used as constraints during the generation of the caption.

<table>
<thead>
<tr>
<th>SPICE Obj. Attr. Rel. Color Count Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up-Down [2]</td>
</tr>
<tr>
<td>21.4 39.1 10.0 6.5 11.4 18.4 3.2</td>
</tr>
<tr>
<td>Transformer [2]</td>
</tr>
<tr>
<td>21.1 38.6 9.6 6.3 9.2 17.5 2.0</td>
</tr>
<tr>
<td>$\mathcal{M}^2$ Transformer [2]</td>
</tr>
<tr>
<td>22.6 40.0 11.6 6.9 12.9 20.4 3.5</td>
</tr>
</tbody>
</table>

Table 3: Breakdown of SPICE F-scores over various subcategories.

Performance is obtained with three encoding and decoding layers, thus confirming the initial findings on the base Transformer model. As our model can deal with a different number of encoding and decoding layers, we also experimented with non symmetric encoding-decoding architectures, without however noticing significant improvements in performance.

**SPICE F-scores.** Finally, in Table 3 we report a breakdown of SPICE F-scores over various subcategories on the “Karpathy” test split, in comparison with the Up-Down approach [2] and the base Transformer model with three layers. As it can be seen, our model significantly improves on identifying objects, attributes and relationships between objects.

3. Qualitative results and visualization

Figures 2 and 3 show additional qualitative results obtained from our model in comparison to the original Transformer and corresponding ground-truth captions. On average, the proposed model shows an improvement in terms of caption correctness and provides more detailed and exhaustive descriptions.

Figures 4, 5, and 6, instead, report the visualization of attentive states on a variety of sample images, following the approach outlined in Sec. 4.6 of the main paper. Specifically, the Integrated Gradients approach [5] produces an attribution score for each feature channel of each input region. To obtain the attribution of each region, we average over the feature channels, and re-normalize the obtained scores by their sum. For visualization purposes, we apply a contrast stretching function to project scores in the 0-1 interval.

4. Novel object captioning

Figure 1 reports sample captions produced by our approach on images from the nocaps dataset. On each image, we compare to the baseline Transformer and show the constraints provided by the object detector. Overall, the $\mathcal{M}^2$ Transformer is able to better incorporate the constraints while maintaining the fluency and properness of the generated sentences.

Following [1], we use an object detector trained on Open Images [6] and filter detections by removing 39 Open Images classes that contain parts of objects or which are seldom mentioned. We also discard overlapping detections by removing the higher-order of two objects based on the class hierarchy, and we use the top-3 detected objects as constraints based on the detection confidence score. As mentioned in Sec. 4.5 and differently from [1], we do not consider the plural forms or other word phrases of object classes, thus taking into account only the original class names. After decoding, we select the predicted caption with highest probability that satisfies the given constraints.

If you need any further assistance or have additional details you would like to include, feel free to let me know!
Figure 2: Additional sample results generated by our approach and the original Transformer, as well as the corresponding ground-truths.
Figure 3: Additional sample results generated by our approach and the original Transformer, as well as the corresponding ground-truths.
Figure 4: Visualization of attention states for sample captions generated by our $M^2$ Transformer. For each generated word, we show the attended image regions, outlining the region with the maximum output attribution in red.
Figure 5: Visualization of attention states for sample captions generated by our $\mathcal{M}^2$ Transformer. For each generated word, we show the attended image regions, outlining the region with the maximum output attribution in red.
Figure 6: Visualization of attention states for sample captions generated by our $M^2$ Transformer. For each generated word, we show the attended image regions, outlining the region with the maximum output attribution in red.
References


