# Final Project Deep Learning 0510725502

Keren Gaiger , Maria Holodov


## I. Abstract

*Transformer-based architectures represent the state of the art in sequence modeling tasks like machine translation and language understanding. Recently, their applicability to multi-modal contexts like image captioning has been largely explored and proved to perform better than other architectures. At this work, we used a promising transformer-based architecture, a Meshed Transformer with memory for Image Captioning and improved its architecture in terms of training-time and computation savings. Those improvements achieved by integrating several key concepts of DeLighT - Deep and Light Transformer, a smaller but deeper architecture, reported as one which gain same or better results than the original transformer. In addition, we investigate the usage of additional input features, pre-trained by the VC R-CNN work. Those features were publicly published, in order to enrich inputs from images to train models that tries to understand the image such as caption creation or question answering. Adding those features improved the results reached by our light transformer and can potentially improve the performance of the original $M^2$-transformer, might be interesting to test this assumption with additional computational resources.*


## II. Introduction

Image captioning, automatically generating natural language descriptions according to the content observed in an image, is an important part of scene understanding, which combines the knowledge of computer vision and natural language processing. It amounts to mimicking the remarkable human ability to compress huge amounts of salient visual information into descriptive language and is thus an important challenge for machine learning and AI research. Some of the many applications of image captioning:

- Helping visually impaired people ”see” the all scene.
- Image efficient indexing- from large data sets, or from videos.
- CCTV camera image indexing and real time alert for malicious activities.
- Better Scene understanding for machines / robots.

A large number of attention-based deep neural networks, such as transformers, have been proposed for image captioning. Those networks are known by their large amount of parameters and mathematical computations and require heavy computational resources. In this work we used one of the most promising transformer architectures ($M^2$-transformer), facing the challenge of training it with limited resources using several key concepts of DeLighT- Deep and Light-Weight Transformer. We managed to drastically reduce the training run time, without harmfully affect the model performance. The second contribution is the interpolation of additional input features which capture the ’visual commonsense’ and the exact visual relationships between objects. While most image captioning systems are good at telling us “what” and ”where”, those input features trying to answer the ”why” ( why the objects located in this manner?, why those object appear together ?). Concatenating them to the ones used in the original $M^2$-transformer paper, resulted in better performance and more readable captions.

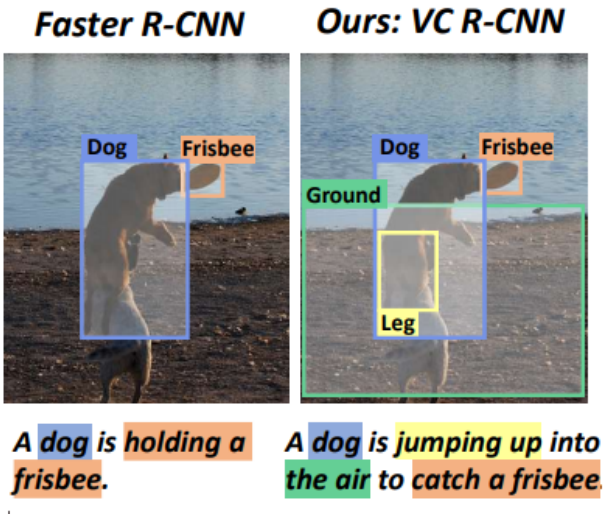



Fig. 1: By adding VC R-CNN features we can be capture more visual relationships and visual attention, being more commonsense aware.

## III. Related Work

A broad collection of methods have been proposed in the field of image captioning in the last few years. With the advent of Deep Neural Networks, most captioning techniques have employed RNNs as language models and used the output of one or more layers of a CNN to encode visual information and condition language generation[18]. Despite their wide adoption, RNN-based models suffer from their limited representation power and sequential nature. After the emergence of Convolutional language models, which have been explored for captioning as well, new fully-attentive paradigms [4] [5] have been proposed and achieved state-of-the-art results in machine translation and language understanding tasks. Likewise, some recent approaches have investigated the application of the transformer model [4] to the image captioning task. In a nutshell, the transformer comprises an encoder made of a stack of self-attention and feed-forward layers, and a decoder which uses self-attention on words and cross attention over the output of the last encoder layer.

Several RNN-based models were examined as part of our research such as the Up-down model [1], GCN-LSTM [17] and AoANet [7], however, when training and testing the models on the COCO dataset, the transformer based models performed better than all of them (see result table I). The transformer-based models we considered were ORT[6], which used a plain transformer and SGAE[16] which used the transformer architecture to compute the attention weights but also incorporated geometric relations between detected input objects to scale those attention weights. The last work considered was the Meshed-Memory ($M^2$) Transformer [3] which introduced two new techniques: (i) encoding a priori knowledge by using persistent memory vectors. (ii) using mesh-like connections between encoding and decoding layers instead of having just a single input from the visual modality (see figures 2 3). $M^2$-transformer showed the best results and was a good candidate for our addition of high-level features, as will be described later in this section. Nevertheless, the $M^2$-transformer is very large and requires massive computational resources which are not necessarily needed. When researching the field of improving transformer architectures, DeLighT- Deep and Light-Weight Transformer [11] was found as a promising model, as it delivers similar or better performance with significantly fewer parameters and operations. DeLighT used a deep and light-weight expand-reduce transformation, that enables learning wider representations efficiently and replacing multi-head attention with single-head attention. Overall, DeLighT networks are 2.5 to 4 times deeper than standard transformer models and yet have fewer parameters and operations. At the aforementioned work of $M^2$-transformer, the encoder is fed with a set of detected object regions extracted from an input image, that were pre-trained by a Bottom-Up and Top-Down Attention [1] mechanism and were publicly published for further use and research. To enrich this input, we used additional image regions, produced by the VC R-CNN model [15], a novel unsupervised feature representation learning method. Given a set of detected object regions in an image (e.g., using Faster R-CNN), the proxy training objective of VC R-CNN is to predict the contextual ,high-level, common sense, objects of a region (see figure 1). Those high-level features, were provided for most of the COCO dataset images and publicly published by the VC R-CNN authors. We concatenated them along with the original visual features used to train the $M^2$-transformer and got better results.

The remainder of the paper is organized as follows: section IV encapsulates the methods we used, section V describes the dataset we trained the model on, followed by evaluation metrics (section VI), training details (section VII), experiments and results (section VIII) and conclusions (section IX). The appendix (section X) elaborates about changes and improvements we did in the project code.

## IV. Methods

This section depicts the $M^2$-transformer model structure and the way we integrated DeLighT to reduce its size. We also describe how we concatenated the high-level features pre-trained by VC R-CNN to improve results.

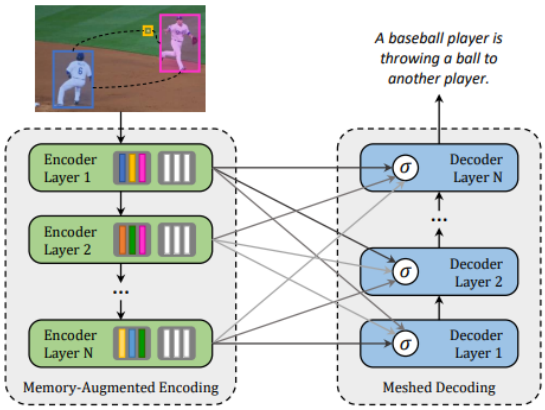



Fig. 2: This image captioning approach encodes relationships between image regions exploiting learned a priori knowledge. Multi-level encodings of image regions are connected to a language decoder through a meshed and learnable connectivity

TABLE I: results of several image captioning models

| Models | $BLEU1$ | $BLEU4$ | $ROUGE$ | $CIDEr$ |
|---|---|---|---|---|
| Up-down [1] | 79.8 | 36.3 | 56.9 | 120.1 |
| GCN-LSTM [17] | 80.5 | 38.2 | 58.3 | 127.6 |
| ORT [6] | 80.5 | 38.6 | 58.4 | 128.3 |
| AoANet [7] | 80.2 | 38.9 | **58.8** | 129.8 |
| SGAE [16] | **80.8** | 38.4 | 58.6 | 127.8 |
| $M^2$-transformer [3] | **80.8** | **39.1** | 58.6 | **131.2** |

### *$M^2$-transformer*

The $M^2$-transformer model can be conceptually divided into an encoder and a decoder modules, both made of stacks of attentive layers (We used 3 layers). While the encoder is in charge of processing regions from the input image and devising relationships between them, the decoder reads from the output of each encoding layer (see figure 2) to generate the output caption word by word. The attention mechanism is implemented in both the encoder and decoder and between them. It operates on three sets of vectors, namely a set of queries $Q$, keys $K$ and values $V$ , and takes a weighted sum of value vectors according to a similarity distribution between query and key vectors. The operator can be formally defined as follows:

$$\mathrm{Attention(Q, K, V)} = \mathrm{softmax}(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d}}})\mathrm{V}, \quad (1)$$

where Q is a matrix of $n_q$ query vectors, K and V both contain $n_k$ keys and values, all with the same dimensionality, and $d$ is a scaling factor.

*1) Memory-Augmented Encoder:* Given a set of image regions $X$ extracted from an input image, the queries $Q$, keys $K$ and values $V$ are obtained by linearly projecting the input features, and the operator can be defined as:

$$\mathrm{S(X)} = \mathrm{Attention(W_q X, W_k X, W_v X)} \quad (2)$$

where $Wq$, $Wk$, $Wv$ are matrices of learnable weights. The output of the self-attention operator, $S(X)$, is a new set of

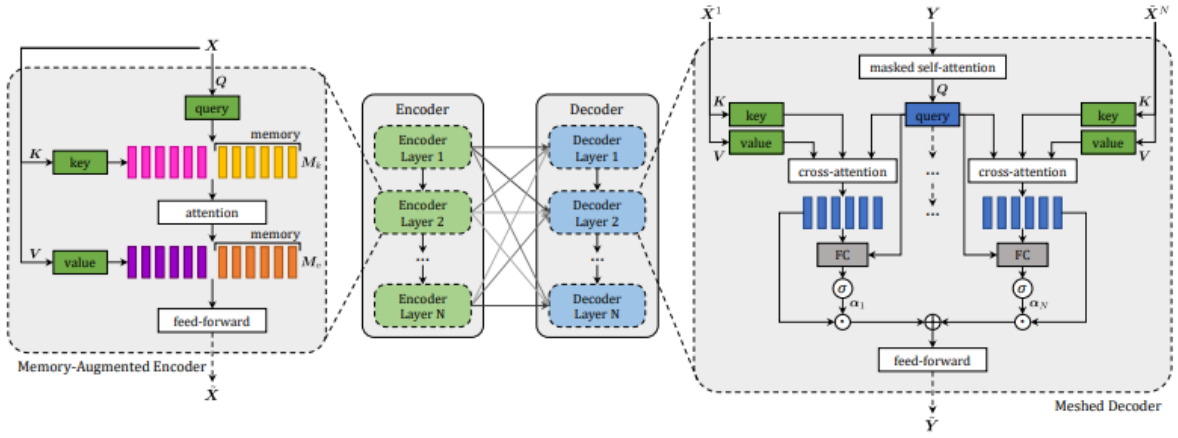



Fig. 3: $M^2$ transformer model

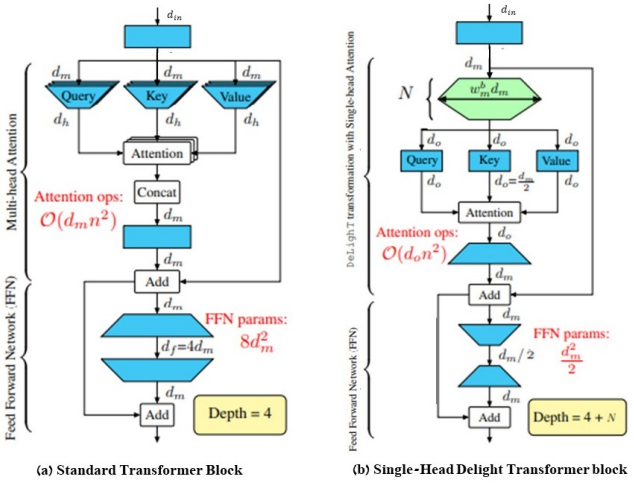



Fig. 4: (a, b) Block-wise comparison between the standard transformer block of Vaswani et al. (2017) and our transformer block inspired by DeLighT

elements, with the same size as $X$. The self-attention operator can be seen as a way of encoding pairwise relationships inside the input set. When using image regions (or features derived from image regions) as the input set, the attention operator can naturally encode the pairwise relationships between regions. However, as it capture solely the pairwise similarities, self-attention cannot model a priori-knowledge on relationships between image regions. For example, given one region encoding a $dog$ and a region encoding a $frisbee$, it would be difficult to infer the concept of $catchgame$ without any a priori knowledge. Again, given regions encoding eggs and toasts, the knowledge that the picture depicts a breakfast could be easily inferred using a priori knowledge on relationships. A different approach to create such prior knowledge is the VC R-CNN feasters that will be explained below.

One way approaching this problem is by **Memory-Augmented Attention**: the set of keys and values used for self-attention is extended with additional "slots" which can encode a priori information. To stress that a priori information should not depend on the input set $X$, the additional keys and values are implemented as plain learnable vectors which can be directly updated via SGD. Formally, the operator is defined as:

$$\mathrm{M_{mem}(X) = Attention(W_q X, [W_k X, M_k], [W_v X, M_v]),} \tag{3}$$

where $M_k$ and $M_v$ are learnable matrices, that on the intuitive level represent some global regions knowledge.

The output from the memory-augmented operator is added to the input via a residual connection and being normalized, then applied to a position-wise feed-forward layer composed of two affine transformations with a single non-linearity, which are independently applied to each element of the set. The FF layer is followed by an additional residual connection and normalization step (see figure 4 (b)). The whole transformer block can be formally defined as:

$$\begin{aligned} \mathrm{F(X)_i} &= \mathrm{U}\sigma(\mathrm{VX_i + b}) + \mathrm{c}, \\ Z &= AddNorm(M_{mem}(X)), \\ \tilde{X} &= AddNorm(F(Z)), \end{aligned} \tag{4}$$

where $X_i$ indicates the $i_{th}$ vector of the input set, and $F(X)_i$ the $i_{th}$ vector of the output. Also, $\sigma$ is the ReLU activation function, $V$ and $U$ are learnable weight matrices, $b$ and $c$ are bias terms. $AddNorm$ indicates the composition of a residual connection and of a layer normalization.

**Full encoder** includes multiple encoding layers stacked in sequence, so that the $i_{th}$ layer consumes the output set computed by layer $i1$. This amounts to creating multi-level encodings of the relationships between image regions, in which higher encoding layers can exploit and refine relationships already identified by previous layers, eventually using a priori knowledge. A stack of $N$ encoding layers will therefore produce a multilevel output $\tilde{X} = (X^1, ..., X^N)$, obtained from the outputs of each encoding layer

*2) Meshed Decoder:* The decoder used in Meshed Cross-Attention, can take advantage of all encoding layers during the generation of the sentence. Given an input sequence of vectors $Y$ , and outputs from all encoding layers $X$, the

Meshed Attention operator connects $Y$ to all elements in $X$ through gated cross-attentions. Instead of attending only the last encoding layer, we perform a cross-attention with all encoding layers. These multi-level contributions are then summed together after being modulated. Formally, our meshed attention operator is defined as

$$\mathrm{M_{mesh}}(\tilde{\mathrm{X}}, \mathrm{Y}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \odot \mathrm{Attention}(\mathrm{W_q Y}, \mathrm{W_k}\tilde{\mathrm{X}}^{\mathrm{i}}, \mathrm{W_v}\tilde{\mathrm{X}}^{\mathrm{i}}), \tag{5}$$

where $\alpha_i$ is a matrix of weights having the same size as the cross-attention results. Weights in $\alpha_i$ modulate both the single contribution of each encoding layer, and the relative importance between different layers. These are computed by measuring the relevance between the result of the cross attention computed with each encoding layer and the input query, as follows:

$$\alpha_{\mathrm{i}} = \sigma(\mathrm{W_i}[\mathrm{Y}, \mathrm{A}(\mathrm{X}^{\mathrm{i}}, \mathrm{Y})] + \mathrm{b_i}) \tag{6}$$

$\sigma$ is the sigmoid activation, $W_i$ is a $2dd$ weight matrix, and $b_i$ is a learnable bias vector. **Full decoder** is built in a multi-head fashion. As the prediction of a word should only depend on previously predicted words, the decoder layer comprises a masked self attention operation which connects queries with keys and values obtained from the left-hand. Also, the decoder layer contains a position-wise feed-forward layer and all components are encapsulated within AddNorm operations. The final structure of the decoder layer can be written as:

$$\begin{aligned} \mathrm{Z} &= \mathrm{AddNorm}(\mathrm{M_{mesh}}(\tilde{\mathrm{X}}, \mathrm{AddNorm}(\mathrm{S_{mask}}(\mathrm{Y})))) \\ \tilde{Y} &= AddNorm(F(Z)), \end{aligned} \tag{7}$$

where Y is the input sequence of vectors and $S_{mask}$ indicates a masked self-attention over time. Finally, our decoder stacks together multiple decoder layers, helping to refine both the understanding of the textual input and the generation of next tokens. Overall, the decoder takes as input word vectors, and the $t-th$ element of its output sequence encodes the prediction of a word at time t + 1, conditioned on $Yt$. After taking a linear projection and a softmax operation, this encodes a probability over words in the dictionary.

### A. DeLighT

Instead of using the suggested transformer block of the $M^2$-transformer model (Figure 4a) , we used the DeLighT transformation, which maps the $d_m$ dimensional input vector into a high dimensional space (expansion) and then reduces it down to a $d_o$ dimensional output vector (reduction) using N layers (figure 4b). It is well known that a standard approach to increase the expressitivity and capacity of transformers is to increase the input dimensions, $d_m$. However, increasing $d_m$ linearly also increases the number of operations in multihead attention, $(O(n^2 d_m)$, where n is the sequence length in a standard transformer block. In contrast, to increase the expressitivity and capacity of the DeLighT block, the depth and width of its intermediate transformations are increased using expansion and reduction phases. This enables the usage of smaller dimensions for computing attention, requiring fewer operations.

Figure 4b presents the transformer block inspired by DeLighT. Inputs are fed into a FF layer and transformed into the size of $d_m$. The $d_m$-dimensional tensors are then fed to the DeLighT reduce-expand transformation to produce $d_o$-dimensional outputs, where $d_o < d_m$. These $d_o$-dimensional outputs are then fed into a single head attention, expanded back to a $d_m$ dimension and fed into the position-wise FF layer, which is the same as in the $M^2$-transformer.

The ability of DeLighT to learn wider representations allows us to replace multi-head attention with single-head attention. The computational costs for computing attention in the standard transformer and the DeLighT block are $O(d_m n^2)$ and $O(d_o n^2)$ respectively, where $d_o < d_m$. Therefore, the DeLighT block reduces the cost for computing attention by a factor of $d_m/d_o$. The last compomont of the DeLight block is the **Light-weight FFN**: Similar to FFNs in transformers, this block also consists of two linear layers. Since the DeLighT block has already incorporated wider representations using the DeLighT transformation, it allows us to invert the functionality of FFN layers in the transformer. The first layer reduces the dimensionality of the input from $d_m$ to $d_m/r$ while the second layer expands the dimensionality from $d_m/r$ to $d_m$, where $r$ is the reduction factor. In the standard transformer, the FFN dimensions are expanded by a factor of 4 ($d_f = 4d_m$. In our experiments, we used r = 4. Thus, the light-weight FFN reduces the number of parameters in the FFN by 16×.

### B. VC R-CNN - Visual Commonsense R-CNN

At this part we are going to describe the model used to create the extra high-level features we concatenated to the original input of the $M^2$-transformer model in order to enrich it. VC R-CNN uses Region-based Convolutional Neural Network (R-CNN) [54] as the visual backbone, and the causal intervention as the training objective. As opposed to the passive observation $P(Y|X)$: "How likely I see person if I see toilet", we keep asking "Why does seeing toilet eventually cause seeing person?" by using $P(Y|do(X))$, which is estimated as follows:

$$\mathrm{P(Y|X)} = \sum_{\mathrm{z}} \mathrm{P(Y|X,z)P(z)} \tag{8}$$

To implement the theoretical idea in the equation above, we propose the proxy task of predicting the local context labels of $Y$'s RoI. For the confounder set $Z$, since we can hardly collect all confounders in real world, we approximate it to a fixed confounder dictionary $Z = [z_1, ..., z_N]$ in the shape of $Nd$ matrix for practical use, where $N$ is the category size in dataset (80 in MSCOCO) and $d$ is the feature dimension of RoI. Each entry $z_i$ is the averaged RoI feature of the $i-th$ category samples in dataset. The feature is pre-trained by Faster R-CNN. Specifically, given $X$'s RoI feature $x$ and its contextual $Y$'s RoI whose class label is $y_c$, the equation can be implemented as $\sum_z P(y^c|x,z)p(z)$. The last layer of the network for label prediction is the softmax layer to get the probability for each one of the $N$ classes. Each two

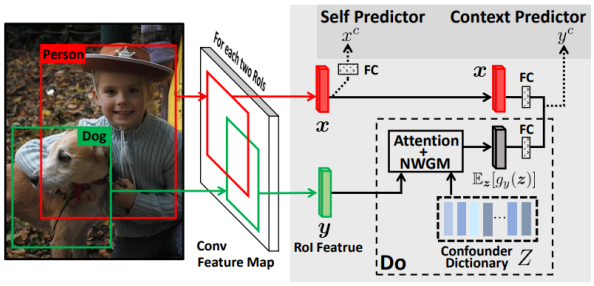



Fig. 5: Overview of VC R-CNN. Any R-CNN backbone can be used to extract regions of interest (RoI) on the feature map. Each RoI is then fed into two sibling branches: a Self Predictor to predict its own class, $x^c$ and a Context Predictor to predict its context labels,$y^c$. The architecture is trained with a multi-task loss.

RoI features $x$ and $y$ branch into two sibling predictors: Self Predictor with a fully connected layer to estimate each object class and Context Predictor with the approximated do-calculus to predict the context label as can be seen in figure 5.

the overall mulit-task loss for each RoI $X$ is:

$$\mathrm{L(X)} = \mathrm{L_{self}(p, x^c)} + \frac{1}{\mathrm{K}} \sum_{\mathrm{i}} \mathrm{L_{ext}(p_i, y_i^c)}, \tag{9}$$

where $L_{self}(p, x^c) = -log(p[x^c])$ is the self predictor loss and $L_{ext}(p_i, y_i^c = -log(p_i[y_i^c])$ is the context predictor loss.

This model was trained in an unsupervised fashion on visual features, based on the assumption that most of the common sense data will be be available in the image captions. For example, image captions unlikely to point out that "people are walking with legs" or "people are eating using forks", this is a "common sense" data that the viewer uses but probably won't mention in the caption. It will be interesting to see if those features will add new knowledge to the M2 transformer model that also learns constant memory features.

## V. Dataset

The dataset we used is MS-COCO Detection [10], which is a popular benchmark dataset for classification, detection and segmentation. It contains more than 120,000 images, each of them annotated with 5 different captions. We follow the splits provided by Karpathy et al. [8], where 5,000 images are used for validation, 5,000 for testing and the rest for training. We also used the VC R-CNN feature vectors for those images, Those images weren't part of the VC R-CNN [15] training set .

## VI. Evaluation Metrics

Before performing any of the evaluation described below, we mapped all the sentences (both the produced and the reference ones) to their stem or root forms. Then, we evaluate the quality of the produced sentences using three standard evaluation metrics.

### A. BLEU - Bilingual Evaluation Understudy Score

This approach, proposed by Kishore Papineni et al. [12], works by counting matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. The comparison is made regardless of word order, where a perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0.

### B. CIDEr - Consensus-based Image Description Evaluation

CIDEr estimates for each image $\boldsymbol{I}_\mathrm{i}$, how well a candidate sentence $\boldsymbol{c}_\mathrm{i}$ matches the consensus of a set of image descriptions $Si = s_{i1}, ..., s_{im}$ [14]. This measure of consensus would encode how often n-grams in the candidate sentence are present in the reference sentences. N-grams that commonly occur across all images in the dataset are given lower weight, using the Term Frequency Inverse Document Frequency (TF-IDF) weighting.

$CIDEr_n$ score for n-grams of length $n$ is computed using the average cosine similarity between the candidate sentence and the reference sentences, which accounts for both precision and recall.

$$\begin{aligned} \mathrm{CIDEr_n}(c_i, S_i) &= \frac{1}{m} \sum_j \frac{g^n(c_i) g^n(\dot{s}_{ij})}{||g^n(c_i)||g^n(s_{ij})}, \\ \mathrm{CIDEr}(c_i, S_i) &= \sum_{n=1}^{N} CIDEr_n(c_i, Si) \end{aligned} \tag{10}$$

A version of CIDEr named CIDErD is available as a part of MS COCO evaluation server to enable systematic evaluation and benchmarking.

### C. ROUGE - Recall-Oriented Understudy for Gisting Evaluation

ROUGE [9] is a set of measures to automatically determine the quality of a summary. We used ROUGE-L which measures the length of the longest common sub-sequence (LCS) between the model candidate and the reference sentence.

## VII. Training Details

We pre-trained our model with a word-level cross entropy loss (XE) and fine tuned the sequence generation using reinforcement learning. When training with XE, the model is trained to predict the next token given previous ground-truth words; in this case, the input sequence for the decoder is immediately available and the computation of the entire output sequence can be done in a single pass, parallelizing all operations over time. When training with reinforcement learning, we used beam search [1]. To decode, we sample the top-k words from the decoder probability distribution at each timestep, and always maintain the top-k sequences with highest probability. As sequence decoding is iterative in this step, the aforementioned parallelism over time cannot be exploited. However, intermediate keys and values used to compute the output token at time $t$ can be reused in the next iterations. This can save a lot of time, but demands more

space in the GPU. CIDEr-D score was used as reward, as it well correlates with human judgment (as mentioned above). We trained our model on n1-highmem-8 (8 vCPUs, 52 GB memory) VM instance provided by Google-Cloud-Platform, with 1 GPU (NVIDIA Tesla K80). We trained it with a batch size of 50, maximum 200 epochs and an early stopping mechanism with patience set to 3. We used NLL loss and the Adam optimizer, starting from a learning rate of 1 which is being scheduled following the strategy of [2].

## VIII. Experiments and Results

We performed two main experiments, each with a different kind of input features:

- Bottom-up-top-down features: 2048-dimensional feature vector for every image region, pre-trained by [1].
- Bottom-up-top-down features + VC-R CNN features: extra 1024-dimensional vector for every image region, pre-trained by [15], were concatenated to the Bottom-up-top-down features.

Both sources published those pre-trained features for further use and research.

Both experiments were tested on three transformer architectures:

1) Baseline - reduced-size version of M2 transformer. $d_k$, $d_v$ were set to 8 with a single-head attention and $d_m$ was set to 64. $d_f$ was set to 1024.
2) DeLighT block with 1 expansion-reduction FF layer (of size 1024).
3) DeLighT block with 2 expansion-reduction FF layers (of sizes 1024 and 512, respectively).

In all experiments, number of blocks was set to 3. For the last two, $d_k$, $d_v$ were set to 64, $d_m$ and $d_o$ were set to 512 and 256, respectively. The $d_f$ was set to 128 ($d_m$ / 2). Figure 6 present the visual difference between the configurations.

The results are presented at table II. As we can see, the baseline of running the compressed version of the $M^2$-transformer get the lowest performance. We assume that the $d_k$, $d_v$ and $d_m$ dimensions were too small to store all relevant information and context. Using the DeLighT transformations (with a single layer in the FF) improved results by factor of 2.5 and incorporating the VC R-CNN input features made results even better. However, the addition of a second layer in the FF component did not lead to an improvement.

We can see some random examples of captions that our model creates in figures 7 and 8. We can see that most of the time the model in very good at recognizing objects (giraffes, cats, fence,toilet,shower, etc.), and scenes (city street, kitchen, bathroom). The model also learned to make assumptions about the actions performed in the image (sitting, feeding, riding, hitting), however this is still challenging for our model and we can see some errors 8a ,8c.

TABLE II: Test results of different models

| Models | $BLEU1$ | $BLEU4$ | $ROUGE$ | $CIDEr$ |
|---|---|---|---|---|
| **$M^2$-transformer compressed** | | | | |
| UpDown | 27.7 | 2.69 | 21.55 | 26.5 |
| **$M^2$-transformer DELIGHT N = [1024]** | | | | |
| UpDown | 63.01 | 21.22 | 49.39 | 105 |
| UpDown+VCR-CNN | 66.92 | 23.26 | 51.46 | 114.5 |
| **$M^2$-transformer DELIGHT N = [1024,512]** | | | | |
| UpDown+VCR-CNN | 62.4 | 21.7 | 49.46 | 105.8 |

## IX. Conclusions

The main goal of this project was improving the performance of $M2$ transformer using new input features, encapsulating the context between image regions, thus producing more coherent, richful captions. Due to its large size, the original $M2$ transformer was trained on a powerful GPU (NVIDIA GeForce RTX 2080 Ti) and probably took a long time to train. Retraining that architecture on the new, expanded inputs was impossible due to time and resources limits. Thus, as our first contribution, we integrated the DeLighT transformations, decreasing the dimension of the input to the self-attention calculations and decreasing the total number of parameters while increasing the depth of the model. This, in addition to the improvement in the data loading process and a faster tokenization (explained in the Appendix), achieved X4 improvement in training time. Another way we dealt with our limited GPU memory is adding gradient accumulation, which grant us the ability of working with larger batches in practice. The aforementioned configuration changes enabled us testing the main goal of integrating the VC R-CNN features, eventually lead to better results, mainly reflected in the CIDEr metric, which is the most relevant for our needs because it gives higher score for a more $uncommon$ and $rich$ description of an image. Based on those preferment's we believe that those features can be helpful and improve the original $M2$ transformer as well, an assumption that might be interesting to test in a further research with more powerful resources. Another further research direction might be training the model with more data, for example with images and captions that are abundantly available on the internet like in the recent paper [13] to achieve robustness and better results.

## X. Appendix

Our project is available in https://github.com/MariaHolodov/project-DL_Transformers_DeLighT and heavily based on the original github repository of the $M^2$-transformer (https://github.com/aimagelab/meshed-memory-transformer). The main code expansions and improvements are:

- **Integration to VC R-CNN Features** The addition of VC R-CNN features required matching them to the old bottom-up-top-down features according to the image and ception ids. In addition, the transformer input layer was increased from 2048 to 3071.
- **Integration of DeLighT** We added 2 components in the MultiHeadAttention function creating the transformer block:
  1) Feed forward layers for the expansion-reduction transformation before the self-attention mechanism
  2) Fully connected layer to expand the output of the self-attention back to the original size
- **Improving Running time** While training our model on the VM of Google Cloud, we suffered from very slow

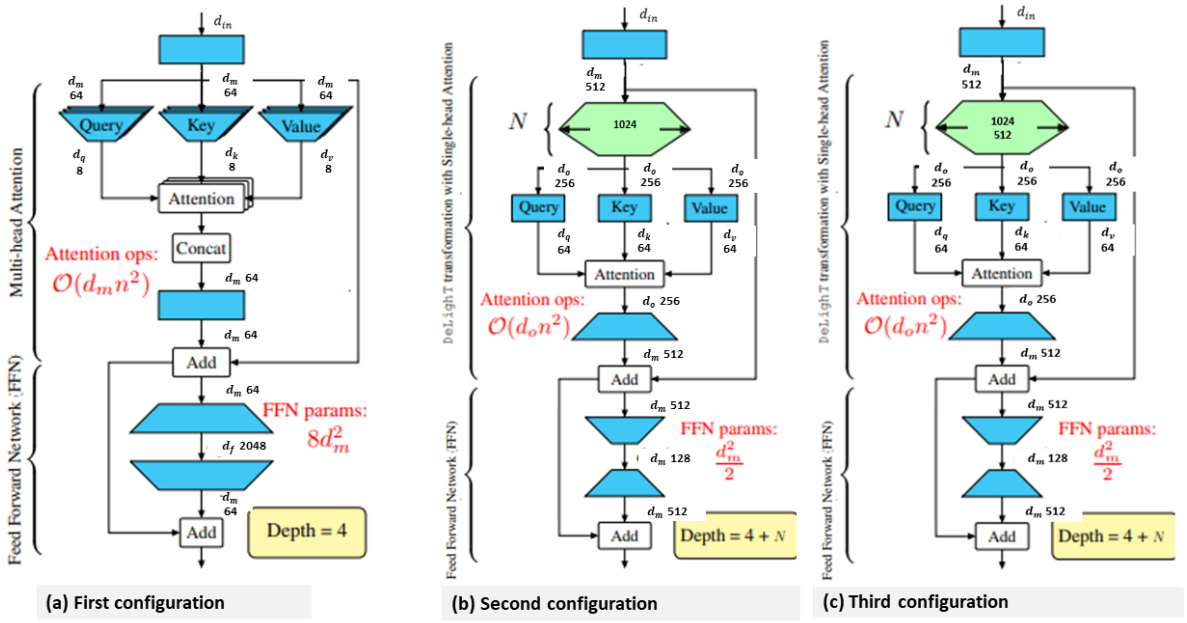



Fig. 6: Three different transformer configurations tested our experiments

training. After Profiling the code we realized that most of the epoch time is spent on data loading from disk (reading an HDF5 file) which means that the GPU isn't exploited efficiently. We changed the data loading mechanism to load all needed data to RAM, only once, in the beginning of the training. That reduced the training time by factor of 4 (!). That change required 8 CPU cores with a total of 52 GB memory.

- **Removing dependencies** The original code used for tokenization the Stanford PTBTokenizer, this involves connecting to Stanford Servers. This process failed many times or took a long time in other occurrences. The writers of [3] tried to solve this by parallelization of this process (using multiprocessing). We changed the tokenization process to work with a standard python package - Natural Language Toolkit (nltk).
- **Accumulation of gradients** Gradient Accumulation is a common technique to deal with limit GPU RAM resources. It means running a configured number of steps without updating the model variables while accumulating the gradients of those steps and then using the accumulated gradients to compute the variable updates. That enabled us to process a batch that fits this limit at each step, while updating model weights based on a larger batch in practice.

To conclude, major changes were done across all parts of the project: data loading and preparation, model architecture and model training. This intervention made us gain a deep understanding of the field, achieved by the various intuitions and concepts we learned in the course.

## REFERENCES

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Łukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems*, 2017.

[3] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10578–10587, 2020.

[4] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *arXiv preprint arXiv:2103.03404*, 2021.

[5] Zhihao Fan, Yeyun Gong, Dayiheng Liu, Zhongyu Wei, Siyuan Wang, Jian Jiao, Nan Duan, Ruofei Zhang, and Xuanjing Huang. Mask attention networks: Rethinking and strengthen transformer. *arXiv preprint arXiv:2103.13597*, 2021.

[6] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. *arXiv preprint arXiv:1906.05963*, 2019.

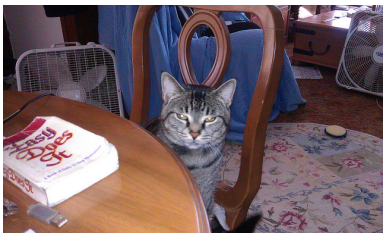
[7] Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. Attention on attention for image captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4634–4643, 2019.

(a) a city street at night with traffic lights

(b) a young boy hitting a tennis ball with a racket

(c) three sheep grazing on grass on a beach

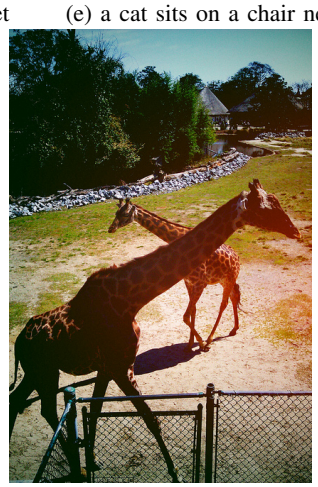
(d) two people riding a motorcycle down the street



(e) a cat sits on a chair next to a table

(f) a person feeding a kitten with a banana

(g) a small bathroom with a toilet and shower



(h) two giraffes are standing next to a fence

(i) a man sitting in a chair in a kitchen

Fig. 7: Example of our captions


[8] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[9] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[11] Sachin Mehta, Marjan Ghazvininejad, Srinivasan Iyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Delight: Deep and light-weight transformer. *arXiv preprint arXiv:2008.00623*, 2020.

[12] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

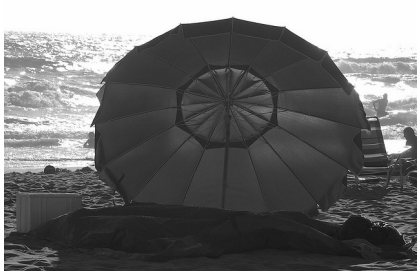
[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[14] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

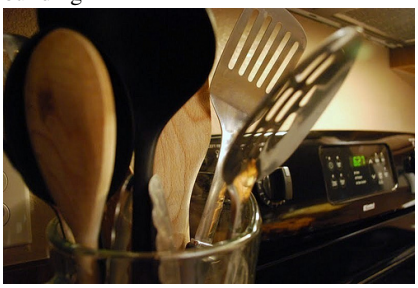
[15] Tan Wang, Jianqiang Huang, Hanwang Zhang, and Qianru Sun. Visual commonsense r-cnn. In *Proceedings*

(a) a young boy sitting on a couch playing a video game controller

(b) a red umbrella sitting on top of a building



(c) a person sitting on the beach holding an umbrella



(d) a metal spoon with a fork sticking out of it

Fig. 8: Example of our captions with errors

*of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10760–10770, 2020.

[16] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10685–10694, 2019.

[17] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 684–699, 2018.

[18] Xuying Zhang, Xiaoshuai Sun, Yunpeng Luo, Jiayi Ji, Yiyi Zhou, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Rstnet: Captioning with adaptive attention on visual and non-visual words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15465–15474, 2021.