applied sciences

MDPI

Article

An Analysis of the Use of Feed-Forward Sub-Modules for Transformer-Based Image Captioning Tasks

Raymond Ian Osolo [1,2], Zhan Yang [2,3,*] and Jun Long [2,3]

[1] School of Computer Science and Engineering, Central South University, Changsha 410083, China; osoloian@csu.edu.cn
[2] Network Resources Management and Trust Evaluation Key Laboratory of Hunan Province, Central South University, Changsha 410083, China; junlong@csu.edu.cn
[3] Big Data Institute, Central South University, Changsha 410083, China
* Correspondence: zyang22@csu.edu.cn

Abstract: In the quest to make deep learning systems more capable, a number of more complex, more computationally expensive and memory intensive algorithms have been proposed. This switchover glosses over the capabilities of many of the simpler systems or modules within them to adequately address current and future problems. This has led to some of the deep learning research being inaccessible to researchers who don't possess top-of-the-line hardware. The use of simple feed forward networks has not been explicitly explored in the current transformer-based vision-language field. In this paper, we use a series of feed-forward layers to encode image features, and caption embeddings, alleviating some of the effects of the computational complexities that accompany the use of the self-attention mechanism and limit its application in long sequence task scenarios. We demonstrate that a decoder does not require masking for conditional short sequence generation where the task is not only dependent on the previously generated sequence, but another input such as image features. We perform an empirical and qualitative analysis of the use of linear transforms in place of self-attention layers in vision-language models, and obtain competitive results on the MSCOCO dataset. Our best feed-forward model obtains average scores of over 90% of the current state-of-the-art pre-trained Oscar model in the conventional image captioning metrics. We also demonstrate that the proposed models take less time training and use less memory at larger batch sizes and longer sequence lengths.

Keywords: image captioning; deep learning; transformers; vision-language

Citation: Osolo, R.I.; Yang, Z.; Long, J. An Analysis of the Use of Feed-Forward Sub-Modules for Transformer-Based Image Captioning Tasks. *Appl. Sci.* **2021**, *11*, 11635. https://doi.org/10.3390/app112411635

Academic Editor: Byung-Gyu Kim

Received: 13 October 2021
Accepted: 4 December 2021
Published: 8 December 2021

1. Introduction

By allowing for the hierarchical representation of features, with complex features described in subsequent layers by successively simpler features, i.e., multiple levels of abstraction, deep learning algorithms have led to many breakthroughs in representation learning dependent tasks. This has resulted in many state-of-the-art achievements in areas such as object detection and recognition, speech recognition, Natural Language Processing, computer vision and vision-language tasks such as image captioning. The fundamental building block of deep learning networks is a neuron which is combined in different combinations, across multiple layers to create more and more powerful networks. This includes simple supervised learning algorithms like Multi-Layer Perceptrons (MLPs), through to more complex architectures such as Recurrent Neural Networks (RNNs) [1], Long Short-Term Memory (LSTMs) [2], Convolutional Neural Networks (CNNs) [3] and Transformers [4]. With the emergence of massive models such as the Bidirectional Encoder Representations from Transformers (BERT) [5], and Generative Pre-trained Transformer (GPT-3) [6], the trend has been to go bigger, more complex and using of more data. The results show that this has been positively rewarding with these systems achieving state-of-the-art results in every field they target.

Recently, works such as [7] have shown that it is possible to achieve competitive results in text classification and translation tasks while using simpler architectures. These architectures alleviate some of the limitations inherent in the more complex ones such as the high computational and high memory requirements that result from, for example, the quadratic nature of self-attention mechanisms that limit sequence length. In our paper, where we use image captioning as proof of concept, we completely discard the self-attention mechanism in the image encoders, and the masked self-attention mechanism in the decoder and introduce a series of feed-forward layers in a regular deep feed-forward framework, and evaluate the most optimum way to represent the input features. We aren't attempting to achieve a new state-of-the-art score, but provide a qualitative analysis of the resulting captions, an empirical analysis of the evaluation metrics, and memory usage, providing a practical insight into the effect of this substitution in vision-language tasks while also demonstrating competitive results with the much simpler architecture.

In image captioning tasks, a machine is designed to automatically generate captions describing an input image. To achieve this, an image feature extractor and a language model are needed. The language model has commonly been implemented by using LSTMs [8–13] and image feature extraction through a state-of-the-art CNN architecture like ResNet [14], VGGNet [15], Inception [16] pre-trained on a large dataset like ImageNet [17]. In this arrangement, the image captioning task is treated as a sequence-to-sequence "translation" task, a technique inspired by the Neural Machine Translation [18]. Due to their sequential nature, which inhibits parallelization and them being slow, alternatives to LSTMs have been explored and implemented resulting in fully convolutional [19,20] and fully attentive architectures such as transformers. First proposed in the paper "Attention is all you need" [4], transformers consist of a stack of self-attention and linear layers in an encoder–decoder architecture. Transformer-based models like [21–25] have achieved state-of-the-art results in image captioning. To achieve this [22–24] use a data-intensive pre-training step to achieve good results. Transformer-only architectures for image captioning like [24] also require massive amounts of pre-training data, and so use a pre-trained encoder from [26], otherwise they report very poor results when training from scratch. In M2 [25], the authors employ a multi-level meshed encoder and decoder increasing complexity and working within the limitations of the quadratic nature of the self-attention technique. Transformers are more computationally expensive, more data hungry and memory intensive architectures.

In order to study possible ways to alleviate some of the above-mentioned problems, while still maintaining a competitive performance, in this paper, we implement and analyze transformer architectures that utilize an encoder composed solely of fully connected feed-forward layers and a decoder that also uses a deep fully connected feed-forward network to encode caption embeddings as shown in Figures 1 and 2. The feed-forward architecture is a simple yet powerful architecture that can adequately model the input textual and image feature representations. The complexity of the feed-forward only encoder section is linear instead of quadratic. In order for it to handle image captioning which involves conditional sequence generation conditioned on the contextual image and textual information, we retain intra-attention only in the cross-attention layer. Furthermore, the input image features used are the output of a Faster R-CNN [27] network, which significantly helps to reduce the dimensions of the input image feature vectors, while retaining and highlighting (via regional proposals) most of its semantic information.

Our main contributions are summarized below:

- We propose transformer-based systems that, in the first stage, utilize a feed-forward encoder using deep fully-connected layers and, in the next stage, a decoder that uses feed-forward layers to encode caption embeddings. We study the memory usage at different batch sizes, the captions generated and the implication of the evaluation metric scores.
- We propose, implement and perform a comprehensive analysis of several combinations of fully connected layers on image captioning models in order to add to the body of knowledge available about the use of simpler networks, in this case, linear layers

in transformer-based vision-language systems, which is an area that has hardly been explored.

- We make a comparative analysis of the use of fully connected layers in vision language models in comparison to self-attention layers and also show that masking of the decoder input features is unnecessary in feed-forward based caption encoding by implementing an unmasked decoder input transformer module.
- We make recommendations and offer tips based on the observations made during the study in order to foster further research in the area of developing simpler transformer models for vision-language applications.
- The MSCOCO benchmark dataset is used to evaluate the performance of the proposed models, which demonstrate competitive results despite the simplicity of the models. The models also show a shorter training time, and lower memory usage at higher batch sizes in spite of containing more parameters. Lower memory requirements and training time allow for use in lower end systems.

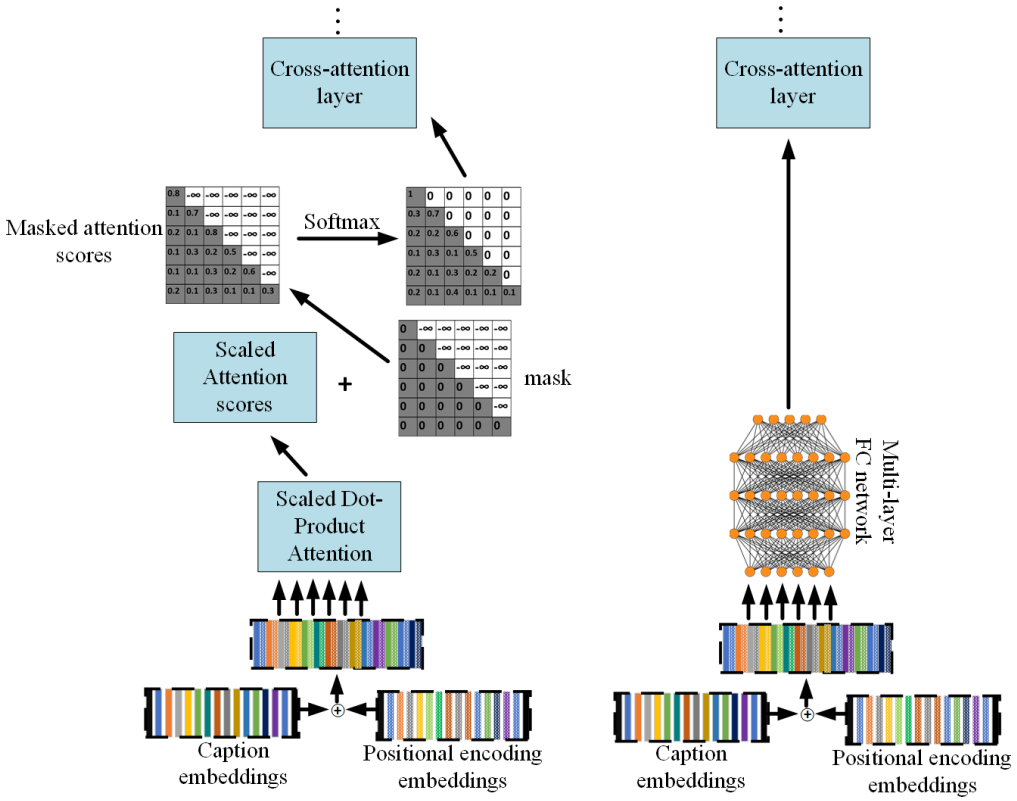


Figure 1. Model complexity reduction: Contrasting the two methods of creating the decoder feature representation that is sent to the cross-attention layer. On the left is the default scaled product attention method that requires masking. The figure on the right shows the much simpler and direct fully connected (FC) layers, with no look-ahead masking or self-attention.

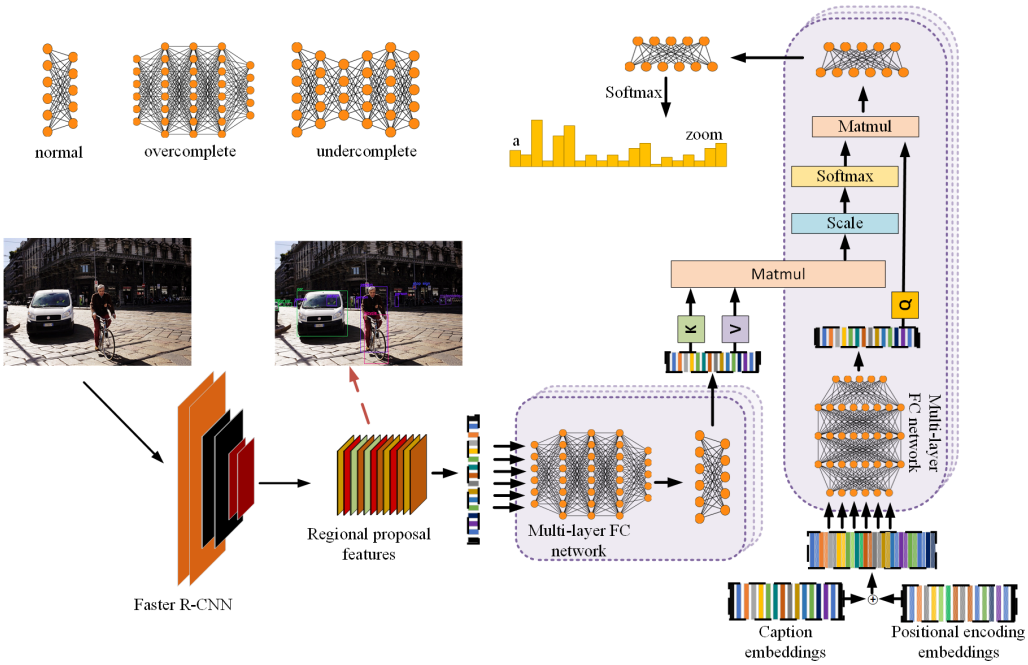


Figure 2. The general framework diagram showing the fully linear encoder using deep fully-connected layers (FC) and a decoder that also uses linear layers to encode caption embeddings. Intra-attention is still maintained in the cross-attention section. The normalization and residual sections are not shown for the sake of brevity.

2. Background

2.1. Algorithms in Deep Learning

There are several deep learning algorithms in use today in research and industry by the computer vision and natural language processing communities. The recent popularity of deep learning algorithms can be attributed to their success in the image recognition field where AlexNet [28], which uses deep convolutional neural networks, achieved a top-5 error rate of 15.3%; 10.8 percentage points higher than the second ranked model, in the ImageNet Large Scale Visual Recognition Challenge [17]. This error rate has been further cut down to 3.5% [29] compared to the human error rate of 5%. CNNs excel at computer vision tasks where they are able to process the spatial and temporal image data dependencies by applying convolutions using filters to create feature maps of different regions of the input image. They are usually applied to tasks such as image classification, image recognition, and image retrieval using powerful architectures such as ResNet [14], and VGGNet [15]. RNNs are used to deal with sequential data. This kind of data requires a network to use previously generated data to make future predictions. They achieve this by incorporating a sort of memory referred to as a hidden state, to save contextual information, that is propagated through the network at each step of a sequence. They are mostly applied as LSTMs or GRUs rather than Vanilla RNNs in tasks such as machine translation [18,30], and automatic summarization [31,32]. MLPs also referred to Artificial Neural Networks (ANNs) and feed-forward networks are the simplest and most versatile of the bunch of deep neural network algorithms. They are comprised of a bunch of neurons in each layer which process the inputs in the forward direction. These neurons can learn a mapping of almost any input data to a required output form with varying degrees of accuracy. They are commonly used to process text and tabular data in applications such as classification and regression prediction. As the subject of this paper, they are discussed in more detail in later sections of this paper. Due to the fact that image captioning involves both images and the text, the two most common deep learning algorithms involved are the CNN to encode the image features into a fixed vector representation, and an RNN to learn to generate the text captions. This is further discussed in the subsequent sections.

2.2. Image Captioning

The quantity and quality of image data generated or used by both users and researchers keeps increasing due to new technologies. This is because of the availability of better image capturing hardware such as smart phones, and the popularity of social media networks where images are uploaded. Companies now need to store more data because of the quality (and by extension, size) of the images being generated. From a computer vision perspective, people are now taking more random images in random places surrounded by random objects, which requires processing by more sophisticated algorithms. This stands in contrast to images in the past, which were taken in studios or well-planned in advance and thus contained more predictable objects. This presents a greater challenge for computer vision tasks such as image captioning. As a result, there are several techniques that have been devised to reduce them to their simplest form while retaining as much of the salient information as possible, for example, by detecting and only retaining the most important features using techniques such as autoencoders and principal component analysis. The other alternative has been to devise more complicated algorithms. As a result, deep learning models have been getting more complicated, bigger and a lot of the state-of-the-art performance of a number of models such as GPT, BERT can be attributed to the amount data used to train them. Some methods using simpler architectures have been proposed to reverse this trend such as using Fourier transforms and fully connected layers [7,33] to get performances close to the state-of-the-art.

Modern image captioning methods are deep learning based, and treat the task as a sequence-to-sequence image-to-text translation task where the image features are input in the encoder in an encoder–decoder architecture which produces a fixed length feature representation vector of the image which is fed to the decoder. The decoder is the language model and is trained in an auto-regressive manner where the image feature vector and previously generated words are used to predict the next word. This technique is usually accomplished by CNN encoders and LSTM decoders [8,9,11–13], CNN encoders and CNN decoders [19,20] and lately transformers [21–23,25].

While transformers have reached several milestones in vision, language and vision-language tasks, they have a lot of shortcomings when implemented in their most popular format as proposed by the original authors [4]. For long sequences as is the case with images and usually with text too, it is very expensive to train transformers both due to the memory and computational requirements. Transformer-only vision or vision-language models [24,34] that do not use convolutional image features as input, require a lot of image training data, as much as needing millions more images [26] to achieve the same results as convolutional based models in tasks such as classification. This is because of the initial unfocused nature of the transformer, which requires it to use some data to learn where to focus, and what to focus on. The inductive biases in convolutional neural networks (due to the kernels) and in recurrent networks (due to their inherent sequential nature) bias them towards an area on which to focus and so they only need to learn what to focus on, and not where to focus. As a result, a number of transformer-based captioning methods vision-language models use pre-training methods [22,23] in order to achieve competitive results.

In this work, we explore the effect of replacing the self-attention layer with fully connected layers and concretely analyze its impact on caption generation. We also suggest a set of guidelines and recommendations based on our observations as pertaining to the aforementioned research. Previous works only concentrate on self-attention [21,25], recurrent [8,9,11] or convolutional [19,20] implementations on the image captioning task, or analyze the scenario in same domain sequence applications [7]. To best of our knowledge there have been no other attempts to explore not using a "look ahead" mask in the masked transformer decoder input module. In among other things, we implement a non-masked fully connected feed-forward decoder input layer, a feed-forward encoder (with non-linear activation functions) module utilizing a number of deep and shallow networks to analyze their effects and show that fully connected layers can be worthy replacements for self-

attention in image captioning. We also analyze the training time and memory consumption at different batch sizes and caption sequence lengths. In addition to image captioning, it should be possible to attain similar benefits in other visual-language tasks such as visual question answering, dialog and vision-language navigation [35–39].

2.3. Masking in Transformer Decoders

In LSTM and GRU sequence generation, their inherent sequential nature ensures that the model only has access to past tokens and the previous hidden state. In transformer-based language models, the text embeddings are input in parallel during the training phase. The decoder is trained in an autoregressive manner, where the previously generated words are used as part of the input to predict the next word. In current implementations of transformer-based text generation tasks, as shown in Figure 1, the self-attention layer over the captions is masked so that the network only has access to previous tokens when learning to predict a future token. According to [4], masking is required because of the parallel nature of inputs to the transformer. Due to this parallelism, the auto-regressive model has access to the full ground truth caption sequence which it may exploit and learn to output the next word in the input sequence, instead of predicting based on the already generated tokens. The authors argue that, at each time step, it will simply look at the input sequence and output the next word, instead of producing a probability distribution from which the next token can be sampled. So in order to achieve this, as in Figure 1, a mask is added to the input sequence representation. The mask is basically a matrix with negative infinity $(-\infty)$ in the position of the elements that you don't want attended to, in order to ensure that they have no influence on the output representation. When the softmax operation is then applied to the resultant matrix, the high scores will be boosted and the negative infinity scores zeroed out. We use fully connected linear layers rather than self-attention to encode both the image feature representation and caption embeddings, without masking the caption input during training as explained in Sections 3.2 and 5 and more formally below.

During the training phase, assume we have an input caption sequence $\mathbf{X} = \{x_1, x_2, x_3, ..., x_n\}$, where $x_1 : n$ are the individual sequence tokens of a sequence of length n. In a sequential model like an RNN, x_1 is given as an input and the model trained to produce x_2. In the next iteration x_1, x_2 are given as inputs to train the model to generate x_3, and so on. As can be seen, the sequential nature prevents the model from looking ahead to see and condition on future tokens. On the other hand, the common transformer architecture [4] uses the full sequence $\{x_1, x_2, x_3, ..., x_n\}$ as input, i.e., passed in parallel. To prevent future tokens from being processed, a look-ahead mask, e.g., $(\{-\infty_1, -\infty_2, -\infty_3, ..., -\infty_n\})$ is added to the input sequence and a softmax operation performed as in (1).

$$\begin{aligned} \mathbf{X} &= \{x_1, x_2, x_3, ..., x_n\} + \{0, -\infty_2, -\infty_3, ..., -\infty_n\} \\ \mathbf{X} &= Softmax\{x_1, -\infty_2, -\infty_3, ..., -\infty_n\} \\ \mathbf{X} &= \{x_1, 0, 0, ..., 0\}. \end{aligned} \tag{1}$$

For subsequent operations, the mask is modified so as to mask the required values, i.e., $\{0, -\infty_2, -\infty_3, ..., -\infty_n\}, \{0, 0, -\infty_2, ..., -\infty_n\}, \{0, 0, 0, ..., -\infty_n\}$ and so on, as required. In this paper we show that when feed-forward layers are used in place of self-attention layers, masking is not required, thereby reducing on the complexity of the model. The idea for this partly arose from the fact that residual connections in a transformer model are used in such a way that the unmasked input information is added to the masked input information in the "Add $ Norm" section of the decoder transformer. The model should still be able to exploit this to learn something about future tokens, thus rendering the benefits of the look-ahead masking procedure not as significant as it is currently assumed.

3. Method

3.1. MLP

A multi-layer perceptron can be described as a mathematical function that maps input values into output values to provide a new representation of the input values. Arranged in different combinations across several layers (deep), they allow for automatic feature engineering, with features learned across multiple levels of increasing abstraction making them capable of tackling complex data. Given a set of N features $\mathbf{X} = \{x_1, x_2, ..., x_N\}$, $\mathbf{X} \in \mathbb{R}^{N \times D}$ where N is the number of features and D is the dimension of each feature, an MLP learns the parameters of a function $f*$ that maps the input features to a target (e.g., a category) y and can be described by $y = \phi(\sum_{N=1}^{N} \mathbf{W}_i \mathbf{X}_i + b) = \phi(\mathbf{W}^T \mathbf{X} + b)$ where $\mathbf{W}$ represents the learnable weights, b the bias and ϕ a non-linear activation function that is used to introduce non-linearities to the network that enables it to be able to map and learn non-linear patterns in the data. In a feed-forward topology (basically a directed acyclic Graph), an MLP consists of an input layer with nodes that correspond to the input feature dimension, one or more hidden layers, and finally an output layer. The neurons in the hidden and output layers compute a weighted summation of the input features and then applies a non-linear activation function. After predicting an output that is compared to the expected output (supervised learning), the loss is computed and the weights updated by back-propagation.

An image is comprised of matrix of pixels. To use this with an MLP, the image will usually be resized, scaled normalized transforming them to have a mean and standard deviation 0.0 and 1.0 respectively. A 28×28 image for example, will be reshaped to a feature vector of size 748 which will be fed to the MLP. In our method, we process the input features using a Faster R-CNN network and use the output as the input features to the encoder. Directly feeding input features of a color image to an MLP would significantly increase the computational complexity of the model since even a simple 64×64 3-channel color image would lead to 12,288 weights to a single neuron. For this reason, we use a 2048-dimensional feature vector from a Faster R-CNN network. The output of the network is a set of 2048-dimensional feature vectors, corresponding to the features detected in each image.

A neural network only understands numbers, typically it is fed a matrix of features. In order to deal with text, every word needs to be represented by a vector. To accomplish this, a vocabulary is defined that contains all the unique words in the input corpus, or at least a subsection of words with a frequency above some arbitrary threshold. Each of the words in the vocabulary can then be represented by a one-hot-encoded vector, and subsequently, each of the words in an input sentence can now be represented by this sparse one-hot-encoded vector. More commonly though, the words are represented by word embeddings. Word embeddings retain the structure of the text, and therefore the context in which they were used. This has the added benefit of identifying similar words and representing them in similar representations within a predefined vector space. A real-valued vector representation of the text is learned using a predefined fixed sized vocabulary learned from a large corpus. To achieve this, each word is mapped to one vector and learned creating a dense vector representation containing hundreds of dimensions that when learned like a neural network captures the meaning of words around them [40].

3.2. MLPs in Image Captioning

Using MLP sub-modules, our objective is to demonstrate and analyze transformer-based vision-language models, specifically, image captioning models. The models we analyze and propose use linear layers to encode and transform the image and caption embeddings rather than using self-attention yet still generate satisfactory captions and evaluation scores comparable to their self-attention-based counterparts.

Transformer models [4], as used in image captioning tasks, consist of two parts: an encoder and decoder section. The encoder creates a fixed vector representation of the input image which is sent to the auto-regressive decoder, which uses this representation together

with the caption embeddings representation to create a probability distribution over the vocabulary. From this probability distribution, the next word in the caption sequence is sampled, i.e., to predict the next token x_{t+1}, we compute the $P(x_{t+1}|I, x_1, x_2, ..., x_t)$ where I represents the image features and $x_1, x_2, ..., x_t$ are the previously generated word tokens.

Unlike the typical models that consist of a self-attention network to compute the relevance or attention score of each image feature in relation to the other features, we use linear transformations in conjunction with activation function non-linearities to learn a representation of the input. The image input features are transformed through one or several fully connected layers as in (2) and (3), respectively.

$$\mathbf{X}_i = \phi(\mathbf{W}^{\mathbf{T}}\mathbf{x}_n + b), \tag{2}$$

where ϕ, $\mathbf{W}$, b are the ReLu activation function, weight and bias of a fully connected layer, $\mathbf{x}_i \in \mathbb{R}^{N_i \times D}$ is the input image sequence with N_i, D being the number of input features and the dimension of each input feature, respectively,

$$\mathbf{X}_i = g(\mathbf{x}_n), \tag{3}$$

where $g(\mathbf{x}_n) = \phi\Big(\mathbf{W}_{(L)}...\phi\Big(\mathbf{W}_{(1)}\mathbf{x}_n + b_1\Big) + b_L\Big)$. L is the layer number in a multi-layer setup.

For the deeper models as in (3), akin to autoencoders, the input features $\mathbf{x}_n$ can be transformed into latent space $h = f(\mathbf{x}_n)$ then the features reconstructed $r = g(h)$. In an under-complete architecture h is of a lower dimension than $\mathbf{x}_n$, whereas it is of a higher dimension in an over-complete architecture.

Image features, as shown in Figure 2 are taken from the output of a Faster-R-CNN model. Each feature $\mathbf{x}_i$ is $\in \mathbb{R}^{N \times D}$ where the dimension of the features $D = 2048$ and N, which is set to 50 is the number of selected features, i.e., regional proposals per image. Because of their source (output of Faster-R-CNN), positional encodings are not used. This is due to the fact that the features are a bag of detections from regional proposals which do not preserve the spatial and geometric location information of each regional proposal. There is thus little to no benefit in using positional encodings in the encoder.

Encoder section: Similar to [4], the stack of encoders is arranged such that the output of one encoder is used as the input of the other next decoder, but different from them, as in [25], instead of only using the output of the final decoder as the final image vector representation, we save the output of each encoder, which will be shared with the corresponding decoder. Consequently, we have the same number of encoders as decoders. The final transformation in the encoder is another linear layer, which is composed of two fully connected layers.

The time complexity is used to depict the amount of computer time required to run an algorithm. The computational complexity of the self-attention mechanism is $\mathcal{O}(n^2 \cdot d)$ whereas that of CNNs and RNNs is $\mathcal{O}(k \cdot n \cdot d^2)$ and $\mathcal{O}(n \cdot d^2)$, respectively, where n is the sequence length and d is the dimension of the features, the k variable in the CNN complexity represents the convolutional kernel size. The feed-forward network complexity is function of the number of layers, number of neurons per layer and the feature dimensions. While the number of neurons in the input layer corresponds to the feature dimension, it may differ for the hidden and output layers. These computations for a fully connected layer are dominated by the matrix operations and are computed as $\mathbf{N}_1 \times \mathbf{N}_2$, where $\mathbf{N}_1 \in \mathbb{R}^{n_1 \times n_2}$ is the dimension of the input matrix and $\mathbf{N}_2 \in \mathbb{R}^{n_2 \times n_3}$ is the dimension of the output matrix. In comparison, as shown in (4), in addition to the quadratic calculation of the relevancy scores, there are also 4 matrix multiplications for the Queries, Keys and Value matrices.

Decoder section: In image captioning applications, captions are both the targets and at the same time one of the inputs. Captions are entered into the decoder in a parallel manner, but usually masked in order to ensure that the decoder can only see the previously generated tokens. In our implementation, where we encode and extract caption information using a linear transformation with ReLu activations, we found that masking

was unnecessary. This is explained in detail in Section 5 and shown in Figure 1. Briefly, as in the standard implementation, caption embeddings are concatenated with sinusoidal positional embeddings. This ordinarily would then go to a masked self-attention layer, but we use a deep fully connected layer to encode these caption features, without masking future captions, i.e., when predicting the token at a position, the model has access to future and past tokens, i.e., a non-masked caption embedding representation with positional contextual information added.

Cross-attention/image-caption translation section: During the training phase, this learns the relationship between the image feature representation and the caption embeddings representation, in an autoregressive manner, so as to produce a new feature representation that can be used to predict the next word in a caption sequence. First off, the linearly transformed caption embeddings from the previous layer are sent to the cross-attention layer. Owing to the fact that this section requires a cross comparison between image features and caption representations, and an attention mechanism where the caption embeddings are used as Queries (Q) over the image feature representations (Keys (K)-Values (V) pair), the possible linear transformations required and attention mechanisms would be more complicated than a self-attention mechanism. We argue that this is the only part of the original transformer architecture that greatly benefits from intra-attention mechanisms. As such, we resolved to use it in only this section of the model. The Query (Q) vectors from the caption embeddings, and the Key (K)-Value (V) vectors from the image feature representation are used to compute the relevance score of each image feature to other features, to produce the attention matrix as shown in (4). See Figure 2 for the visual representation of the previous statement.

$$Attention\,(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})\,V \tag{4}$$

where $Q = X_i * W_Q$, $K = X_i * W_K$, $V = X_i * W_V$. X_i represents the encoder input features $x_1 ... x_n$; W_Q, W_K, W_V are learned projection matrices for the queries, keys and values, respectively. Multi-head attention as well as residual connections, are employed to send information from the input of one layer to the next in order to reduce information loss. This loss may occur due to the many transformations undergone by the data. This also helps the gradients propagate through the model more easily. Since we do not just use the final output of the last decoder, but rather the final outputs of the all the encoders (multi-level encoder-decoder network), are saved for interfacing with the corresponding decoder.

For the objective functions, as with many similar sequence generation works, we first optimize the cross entropy then perform CIDEr-D metric optimization [41], i.e., given $(y^*_{1:T})$ as a sequence of ground truth word tokens where T is the sequence length of the sequence $y_1, ... y_T$. The cross-entropy loss is given by:

$$L(\theta) = -\sum_{t=1}^{T} \log\left(p_\theta\left(y^*_t \mid y^*_{1:t-1}\right)\right) \tag{5}$$

where θ represents the model parameters. The CIDEr score is optimized by SCST:

$$L_{RL}(\theta) = -E(y_{1:T \sim p\theta}[r(y_{1:T})] \tag{6}$$

where r is the score function. The gradient is then approximated by:

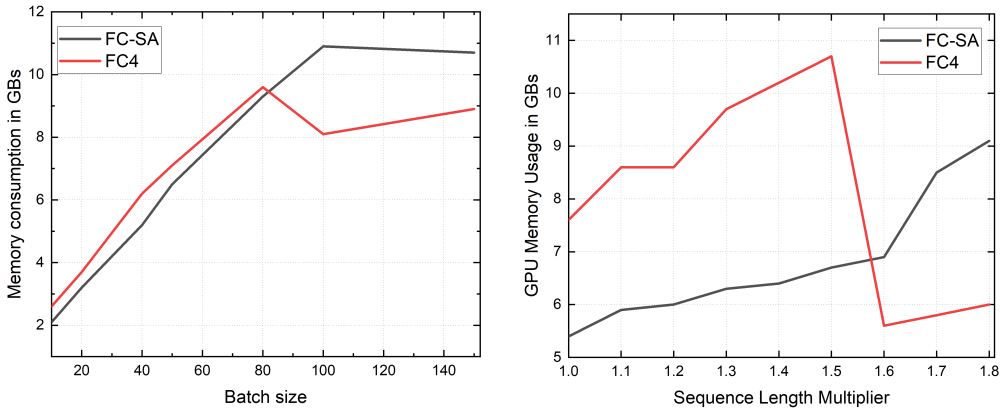
$$\nabla_\theta L_{RL}(\theta) \approx -(r(y^s_{1:T}) - r(\hat{y}_{1:T}))\nabla_\theta \log p_\theta(y^s_{1:T}) \tag{7}$$

Summary: How it all comes together: The output of Faster-R-CNN model, which contains detected image features (regional proposals) is used as the input to the encoder of out transformer model. The first sub-module which is comprised of fully connected layers, learns to create the most optimum generalized representation of the input features during training. The position-wise feed-forward layers create a fixed vector representation

of the input image in the model dimension, which will be the output of the encoder and one part of the decoder input for the cross-attention sub-module. A representation of the captions which are the decoder inputs and targets (outputs) is created using a deep fully connected decoder input sub-module and passed on to the cross-attention sub-module. The cross-attention sub-module detects and learns the relationships between the caption embedding representation and the image feature representation. From the output of the cross-attention layer, transformed by the position-wise linear layer, the image-caption representation is passed through a final linear layer and after a softmax operation outputs a probability distribution over the vocabulary. The first or next token in the caption sequence can be sampled from this probability distribution. During inference this is done in an auto-regressive manner until the end of sequence token is predicted.

4. Experiments and Results

The proposed model is set up as shown in Figure 2 with the memory usage graphs shown in Figure 3 and graphs for evaluation metrics shown in Figure 4. Below we detail some of the information such as implementation settings and datasets that were used to train and test the model. In addition, in the evaluation studies section, we justify the different components that were chosen in the proposed model, and also give information about other experiments we performed on the model variants to validate some of the decisions that we made.



(**a**). Memory usage at different batch sizes compared.

(**b**). Memory usage at different sequence lengths compared

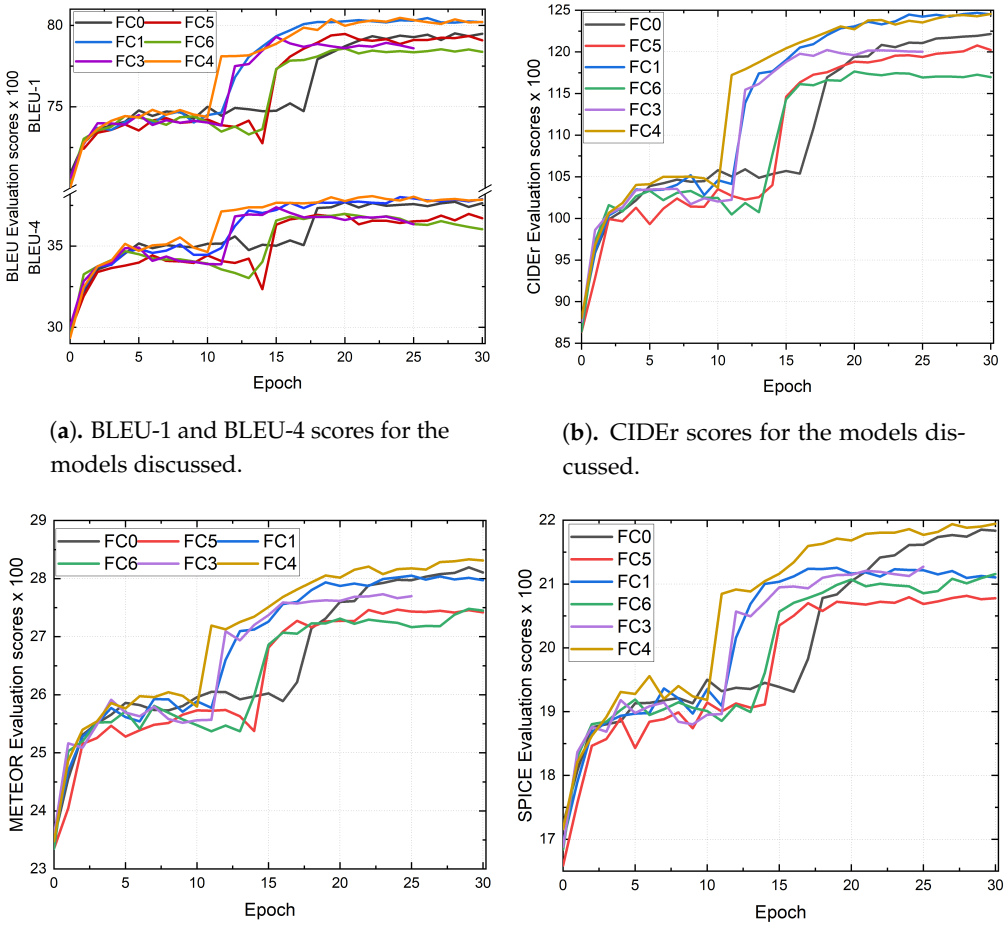
Figure 3. Graphs showing a comparison between the memory usage of the self-attention based variant (FC-SA) and our best model (FC4) at: (**a**) different batch sizes. (**b**) different sequence lengths. The memory usage reported is for usage during the cross-entropy optimization stage.

4.1. Materials: Dataset and Evaluation Metrics

The MSCOCO dataset [42], using the pre-defined Karpathy splits was used for training, validation and testing. Using the pre-defined Karpathy splits makes it easier to make comparisons with other models. The dataset consists of 113 k, 5 k and 5 k images for the training, validation and test sets, respectively. The effectiveness of the proposed methods was quantitatively evaluated using the standard evaluation metrics. The BLEU [43] evaluation metric is an n-gram precision based metric where the "n" represents number of n-grams from 1–4, i.e., BLEU-1 to BLEU-4. The CIDEr [44] evaluation metric uses the Term Frequency-Inverse Document Frequency (TF-IDF) statistical measure to give more weightage to important n-grams. ROUGE [45] is a metric more commonly used in automatic summarization, but can also be used to compare generated captions and the ground truth captions. METEOR [46] performs unigram matching. The SPICE [47] metric calculates an F1-score over caption scene-graph tuples, i.e., computing the balance between the precision and the recall.

4.2. Settings Implementation

A Faster R-CNN model pre-trained on the ImageNet dataset and fine-tuned on the visual genome dataset is used to extract image features. The model was first trained by optimizing the cross-entropy loss, followed by CIDEr-D optimization (REINFORCE). A vocabulary size of 10,000, a beam width of 5, the Adams optimization algorithm, dropout and early stopping were employed. The number of feature detections from the Faster R-CNN network is limited to 50. PyTorch was the deep learning library used, running on Intel E5-2600 CPUs and 2080Ti GPUs.



(**a**). BLEU-1 and BLEU-4 scores for the models discussed.

(**b**). CIDEr scores for the models discussed.

(**c**). METEOR scores for the models discussed.

(**d**). SPICE scores for the models discussed.

Figure 4. The Graphs above show various evaluation metrics across the training epochs for select model variants corresponding to: (**a**) BLEU-1, BLEU-4 (**b**) CIDEr, (**c**) METEOR, and (**d**) SPICE scores.

4.3. Evaluation Studies

We perform memory consumption and training time per epoch tests (results shown in Table 1), and implement several model variants (configurations shown in Table 2 and Figure 5). The evaluation metric results of our best model compared to the comparison models are shown in Table 3 while those of the model variants are displayed in Table 4. Details of the model variants and evaluation studies are discussed below: where **X** refers to the input feature set of either text or image features.

(i) To *evaluate the memory consumption* of our model to a similar one using self-attention, we designed a variant that we refer to as FC-SA. In this variant, the encoder feed-forward network is replaced by a self-attention network, and the decoder one, replaced by a masked self-attention layer. The GPU memory consumption with the batch size

set to 10, 20, 40, 50, 80, 100 and 150 for both our best model (FC4) and the self-attention based variant FC-SA are shown in Figure 3a.

(ii) To *evaluate the memory consumption* at different sequence lengths, just as above, we use the FC4 and FC-SA variants. Since it is just a test of memory consumption and not a test of caption generation, we duplicate the captions to create longer captions, i.e., every caption is a multiple of its original length. So we measure memory usage at multiples of the caption length, i.e., (1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8) * caption length. Basically, given a caption sequence $X = \{x_1, x_2, ..., x_n\} = X * 1.0$, then, $X * 1.5 = \{x_1, x_2, ..., x_n, x_1, x_2, ..., x_{\frac{n}{2}}\}$ where n is the original sequence length. The results are shown in Figure 3b. We also report the inference time required to generate captions for 5000 randomly selected images.

(iii) To investigate the performance of the model before and after using a look-ahead mask, we ran the FC4 and FC-SA variants with and without the mask.

(iv) To evaluate the time complexity of the introduction of the feed-forward layers, we build 2 variants of our model that are comprised of only the encoder. One variant contains the self-attention layers. In the other variant, the self-attention layers are replaced by feed-forward layers. The time taken to produce the image feature representation is noted and shown in Table 1.

(v) FC0, FC1 and FC2 *investigate the effect of change in number of layers or stacks* on the base variant model FC0. FC0 is designed with the "normal" configuration as shown in Table 2 where input features (2048-dimensional) are directly transformed into the model dimension (512-dimensional) and forwarded to the next stage, i.e., position-wise feed-forward stage, without any other changes. The experiments covered 3-, 6- and 12-layer variants. A layer here refers to one whole encoder or decoder section. Varying the number of layers may improve or degrade the performance of the model. While it could initially improve performance because of the additional discriminatory information mined, the increase in the complexity may result in a reduction in performance, and an explosion in the size of the model and training time. This is discussed further in Section 5.

(vi) FC3 is used to investigate *mapping the input image features into a higher dimensional space*. $\mathbf{X} \in \mathbb{R}^{N \times D} \xrightarrow{\text{mapped}} \mathbf{X}' \in \mathbb{R}^{N \times D'}$, where $D' > D$. This is the initial building block for the next couple of experiments. In this experiment, we try to find out the effect of mapping the input image features (2048-dimensional) into a higher dimensional space (4096-dimensional), an arrangement that mimics an over-complete autoencoder. The aim is to see if it creates a better representation of the input features, which would then lead to better captions. In the same line, in FC5, we implement a model that mimics an under-complete stacked auto encoder configuration, i.e., $\mathbf{X} \in \mathbb{R}^{N \times D} \xrightarrow{\text{mapped}} \mathbf{X}' \in \mathbb{R}^{N \times D'} \xrightarrow{\text{mapped}} \mathbf{X}'' \in \mathbb{R}^{N \times D}$, where $D' < D$. Both these are done to try to get the best possible representation of the input features when using feed-forward layers.

(vii) FC4 and FC8 are designed to investigate the *effect of having a deep feed-forward network over the caption embeddings*. The caption embeddings representation is created using a non-masked deep fully connected neural network. One of the novelties of this paper is the use of a non-masked caption embeddings and so we perform experiments on a variant that has "normal" encoder feed-forward layer and deep fully connected decoder input sub-module that contains a series of 2048 dimensional hidden layers before being converted to the model dimension. We also test and report the results of increasing the number of layers in this configuration.

(viii) Building on the previous configuration, we perform experiments with variants FC6 and FC7 where we use a deep feed-forward networks in both the encoder and decoder sections. This is done to see if we can extract any additional benefits from *representing both the images and the text using a series of high dimensional feed-forward layers* in both model sections.

Table 1. The memory consumption in gigabytes (GB) and training time in minutes (Mins) of the feed-forward variant (FC4) compared to the self-attention variant (FC-SA). t-enc refers to the time taken by the encoder to produce the image feature representation. t/E refers training time (t) per cross-entropy optimization Epoch (E).

Batch Size	FC-SA (GB)	FC-SA t-enc (Mins)	FC-SA t/E (Mins)	FC4 (GB)	FC4 t-enc (Mins)	FC4 t/E (Mins)
10	2.1	32	47	2.6	26	42
20	3.2	26	37	3.7	19	34
40	5.2	23	34	6.2	20	32
50	6.5	22	32	7.1	17	28
80	9.3	21	32	9.6	16	30
100	10.9	21	31	8.1	18	31
150	10.7	19	31	9.5	16	31

Table 2. Model configuration for the variants used in the ablation studies. ff = feed-forward, sae = stacked autoencoder, normal = input features directly transformed into the model dimension and forwarded.

Method (Layers)	Encoder	Decoder
FC0 (3), FC1 (6), FC2 (12)	normal	normal
FC5 (3)	sae	normal
FC3 (3)	deep ff	normal
FC6 (3), FC7 (6)	deep ff	deep ff
FC4 (3), FC8 (6)	normal	deep ff

Table 3. Performance of comparison models on the Karpathy MSCOCO splits. B1 = BLEU-1, B4 = BLEU4, M = METEOR, R = ROUGE-L, C = CIDEr, and S = SPICE.

Method	B1	B4	M	R	C	S
Xu et al. [9]	70.7	24.3	23.90	-	-	-
SCST [41]	-	34.2	26.7	57.7	114.0	-
Up-Down [11]	79.8	36.3	27.7	56.9	120.1	21.4
RFNet [48]	79.1	36.5	27.7	57.3	121.9	21.2
FC4	**80.3**	**38.1**	**28.2**	**58.0**	**124.4**	**21.8**

Table 4. Performance of select variant models on the Karpathy MSCOCO splits. All variants' results reported after CIDER-D optimization.

Method	B1	B4	M	R	C	S
FC0	79.3	37.4	**28.2**	57.6	122.0	**21.8**
FC5	79.3	37.0	27.5	56.7	120.8	20.8
FC1	80.2	37.9	28.0	57.4	**124.4**	21.1
FC6	78.4	36.3	27.4	56.7	117.0	21.0
FC3	78.6	36.6	27.6	56.9	120.0	21.1
FC4	**80.3**	**38.1**	**28.2**	**58.0**	**124.4**	**21.8**

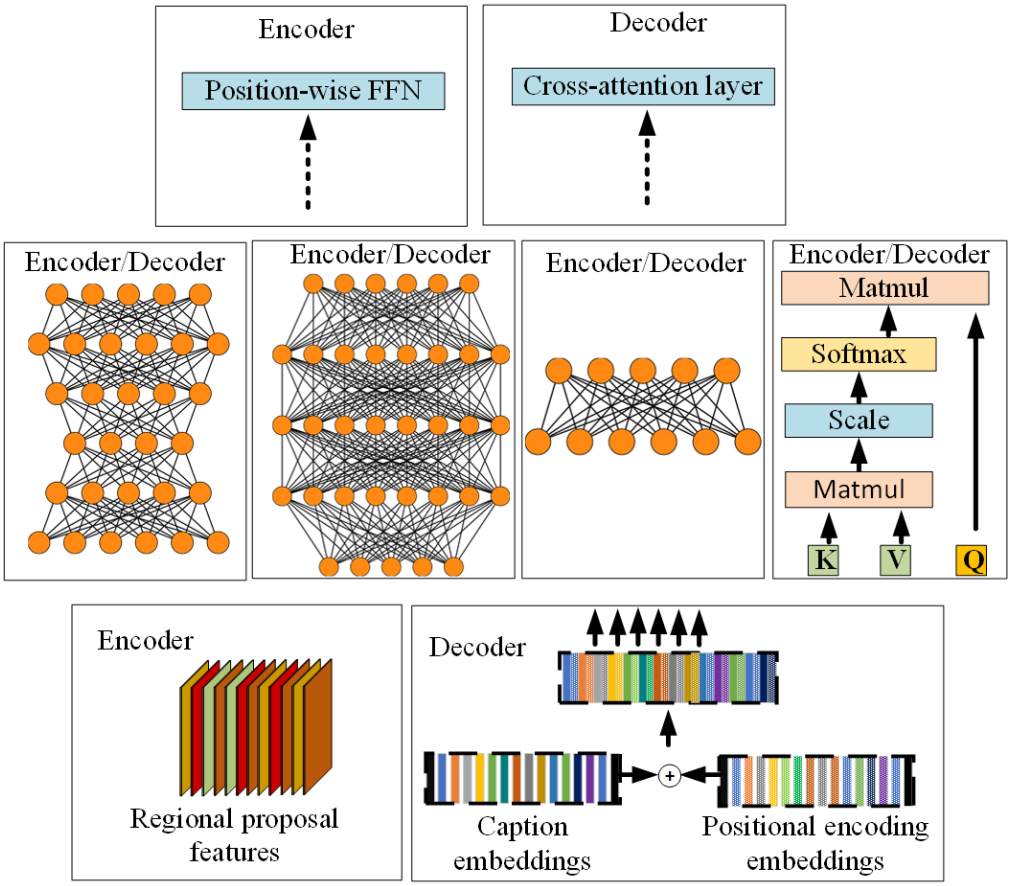


Figure 5. The configuration options available at each stage in the variants tested. The first stage (bottom) shows the formats available for the input features to the model. The second stage contains the options that were considered for learning the feature representation. The top shows the next stage depending on whether it is in the encoder or decoder section. As indicated, some options are only available to the encoder, decoder or to both.

The models' performance, and insights are discussed in Section 5.

4.4. Comparison with Other Models

Xu et al. [9] is the classic state-of-the-art deep learning based image captioning model that used attention and the CNN-LSTM paradigm. SCST [41] introduced the idea of directly optimizing non-differentiable metrics like CIDEr using a policy gradient methods in image captioning. Up-Down [11] integrate grid-based features and region proposal features from a Faster R-CNN network to perform image captioning. RFNet [48] exploit information from multiple encoders to generate informative representations for the decoder. The results from our best model, FC4 are competitive in all metrics and better than the comparison models.

5. Results Discussion

The associated epoch-by-epoch graphs are shown in Figure 4. The evaluation metric scores are displayed in Tables 3 and 4 for the comparison models and the variants, respectively. Table 5 shows a sample of the captions generated by the model variants alongside the ground truth captions (GT). Additional information is displayed in the Appendices A.1 and A.2. Figure A1 in the Appendix A.1 shows additional graphs that highlight some specific comparisons that provide deeper insights into the performance of select model variants. In Appendix A.2, Tables A1 and A2 display a larger sample of captions generated by the linear models explored in this work.

Table 5. Sample captions generated by the models.

GT: a large semi truck pulling a blue tractor behind it.
FC0: a large truck is driving down a traffic light.
FC1: a large dump truck driving down a traffic light.
FC6: a construction truck carrying a crane on the road.
FC4: a construction truck driving down a traffic light on the road

As can be seen in Table 4, FC4 and FC1 are by far the best models overall when the evaluation metric performance, simplicity and training time is taken into consideration. FC1, follows the "normal" configuration shown in Table 2 with 6 encoder–decoder stacks, whereas FC4 simply has a deep fully connected decoder input sub-module. The evaluation result metrics of the 6-layer stacked FC1 that are significantly higher than FC0, its 3-layer counterpart show the merits of increasing the number of layers. This is also clearly observed in the graphs in Figure A1a. Both FC4 and FC1 performed better than the model that had projected input image features into deep higher dimensional spaces (FC3, FC6 and FC7) or where a reconstruction of the most important input features (as in an autoencoder) was attempted (FC5). This is because the input features which are from a Faster R-CNN network are already compressed with the most salient features (regional proposals) highlighted and extracted as feature vectors, making any further such manipulation mostly counterproductive, actually leading to a degradation in performance. Compared to the state-of-the-art Oscar [22] model, which uses self-attention and pre-training (on 6.5 million image-text pairs), FC4 attains 94% (38.1 vs. 40.5), 95% (28.2 vs. 29.7), 90% (124.4 vs. 137.6) and 96% (21.8 vs. 22.8) of the Oscar-big model [22] score in the BLEU-4, METEOR, CIDER and SPICE metrics, respectively. Compared to the Oscar-large model (larger hidden size, see their paper for more details), FC4 attains 91% (38.1 vs. 41.7), 92% (28.2 vs. 30.6), 89% (124.4 vs. 140.0) and 89% (21.8 vs. 24.5) of the model's score in the BLEU-4, METEOR, CIDER and SPICE metrics, respectively. In terms of time required to train, the results of tests that were performed at different batch sizes as shown in Table 1, demonstrate that the fully connected feed-forward variant consistently required less time to train than the self-attention variant.

From the same table, it can be deduced that most of the time is spent in encoder section of the model. For example, at a batch size of 50, 60.7% and 68.8% of the total time is spent in the encoder section for the fully connected compared to the self-attention variant, respectively. This is because that section involves reading to memory several gigabytes of image data compared to a few megabytes of caption data in the decoder section. The time taken by the FC4 model is also much less than the time taken by the self-attention based model. We surmise that this could be because of the much simpler architecture of the feed-forward model which leads to far fewer processing steps. In terms of inference time, in order to process and create captions for 5000 images, the feed-forward based model takes 2.1 min compared to 1.9 min for the self-attention based one. This comes out to an average of about 39.7 compared to 43.9 images per second for the feed-forward and self-attention variant, respectively, on the GPU server. Since the inference part is one that is likely to be processed locally on a desktop pc or laptop, we repeated the test on an Intel i7-6700HQ processor with an Nvidia GeForce-GTX 960M GPU, where it took 16.2 min (5.1 images per second) compared to 15.8 min (5.2 images per second) for the feed-forward and self-attention variant respectively. A large chunk of the processing time for this sequential task is spent in dealing with copying and moving images in memory and so it is not surprising that there is a negligible difference in the time required to process each image for either one of the two networks. The disparity only becomes non-negligible when creating captions for several thousands of images, on a sufficiently powerful machine.

Memory usage: The graphs in Figure 3 show the memory usage of our best model (FC4) compared to a variant that uses self-attention layers (FC-SA). From the Figure 3a it can be observed that the feed-forward variant only uses slightly more memory as the batch sized increase from 10 to 80. At a batch size of 100, there is a sharp change in the memory usage of both models. FC4 shows a sharp decrease in memory usage while the usage for FC-SA stops increasing rapidly, almost flattening out. This is particularly fascinating since it contains about 70% more parameters than the self-attention based counterpart. We think that the 100 batch size created ideal conditions for the GPU we used, wherein, the batch size saturates the CUDA cores and the GPU performance is most efficient. This ideal performance continues to the next tested batch size. The FC4 (feed-forward) variant benefits most, using 25% less memory at that ideal batch size. This shows that the simplicity of the proposed model, containing fewer components reduces on the computational complexity of the setup.

Figure 3b contains a graph that shows the memory consumption of the two compared models (FC4 and FC-SA) at different caption sequence lengths. As can be seen, as the sequence length of the captions increases, the quadratic nature of the self-attention mechanism starts manifesting itself. A repeated observation concerning the feed-forward variant is that at some point, the GPU is able to efficiently represent the linear layers and this results in a significant drop in memory consumption. The self-attention Q-K-V mechanism is also made up of linear layers organized in a matrix form, which also benefits from the GPU efficiency at that ideal sequence length, resulting in a reduction in its memory usage too. This is can be interpreted from the less steep slope. Its quadratic nature still causes it to continue increasing its memory consumption at a faster rate than its feed-forward counterpart. A GPU is very adept at handling the matrix multiplications which constitute most of the work done during the training process. The linear transformations in both variants benefit greatly from this.

Non-masked caption embedding representation: One of the major changes to the network architecture was the utilization of a non-masked fully connected layer to capture the semantic information and token relationships of the caption embeddings in place of intra-attention. In previous implementations of transformer-based text generation tasks, the self-attention layer over the captions is masked so that the network only has access to previous tokens when learning to predict a future token. We do not apply masking in the proposed models and show that our model inference works properly and believe that the model actually benefits from having current and previous information during training. In language generation, the context of a word may not only depend on the words before it, but also those that may come after it. Almost all supervised sequence generation deep learning based models use teacher forcing, and not using masking as in our case is simply an extreme case of teacher forcing. The concern that the deep learning models, which are adept at finding and exploiting any patterns will simply learn to predict the next word in the input distribution by looking at the next word in the sequence is not completely valid, at least, not in our models. The model is still able to learn and capture the semantics in the input representation well enough that it is able to predict the next word at inference time. This is proven by the accurate and relevant captions that are auto-regressively generated during inference, with deeper decoder feed-forward variants performing better at caption generation because they are able to learn to create a better representation of the input caption features. When we performed an explicit comparison between the performance of the model with and without using the look-ahead mask, between our best model, FC4 and its counterpart based on a self-attention mechanism (FC-SA), we discovered inconsistencies between the results. While both models do actually operate with or without the look-ahead mask, the difference was the level of variability between results, which seems to point to the likelihood that using a look-ahead mask does also help reduce on the stochastic nature of the model. Judged solely on the results of the CIDEr evaluation metric, based on only four complete runs of the models (train + test) due to time and computational restraints, there was an average of a 6% variance in the results for the FC-SA compared to

2% for the FC4 model. Over the four runs, the CIDEr scores, without using a look-ahead mask were 127.7, 128.8, 121.1, 123.0 and 122.0, 121.9 124.4, 124.2 for the FC-SA and the FC4 model, respectively. When a look-ahead mask was used, the score varied by less than 1% for both, i.e., 128.7, 128.8, 128.7, 128.8 and 124.3, 124.4, 124.1, 124.4, for the FC-SA and the FC4 model, respectively. These results show that the look-ahead mask is more important in self-attention based mechanisms than it is in feed-forward based mechanisms. It further shows that using it also improved on consistency between the results obtained. From a practical point of view, in terms of captions generated, the difference is much smaller. We believe that this is due to the beam search algorithm that we use, combined with the sampling method that is based on a softmax probability distribution over a predetermined vocabulary set, ensuring that the most appropriate token is chosen. This means that there needs to be a much bigger difference between the evaluation metric scores for the quality of the captions to start decreasing.

Stacked Layers: Increasing the number of layers is not a sure way to improve performance and may be detrimental to the performance of transformer models as shown by [49]. In image captioning, ref. [25] got their best performance when they stacked three encoder and three decoder models, and showed a significant drop in performance when they tried to stack more decoder layers. In our experiments using linear models we actually had mixed results. The models usually performed better, the more modules we stacked, up to a certain point where there was little to no improvement. Our best model was the three-layer FC4 model, but the same model with six layers, performed about the same, in fact marginally worse in some metrics. In the case of the base models (FC0, FC1, and FC2), the six-layer model performed almost the same as the 12-layer model, but both performed much better than the three-layer variant. We also observed the best performance in a six-layer architecture, when we only cross interfaced the top three stacks, with the lower three just processing and sending their outputs to the ones above them. We believe that this is because this increased depth starts to cause optimization issues.

The models that first encoded the image features using deep encoders (FC3, FC5) initially had high scores, especially in the BLEU-4 metrics before dropping. This is because in the earlier stages of the training process, the deep fully connected nature of the model allows it to easily map individual aspects of input features to the captions more efficiently, but as the model gets more refined and the focus turns to mapping an understanding the relationship between the individual feature and the captions, the deep nature becomes a hindrance to image representation. It starts experiencing optimization issues. The captions produced at this point, initially containing accurate but repetitive patterns, start becoming syntactically inaccurate. It does eventually converge, but produces the worst captions. Graphs highlighting the SPICE vs. BLEU-1 scores are shown in Figure A1c. It shows that even with the high number of accurate unigrams generated (BLEU-1), the resulting sentences did not correlate well with human generated captions (SPICE).

Comparison models: When FC4 is compared to the comparison models as in Table 3, the results using the fully connected layers are higher than the compared models which consist of state-of-the-art LSTM and CNN-based models. They are a few points behind the performance of the best transformer models in many of the metrics, but almost at par with them in the unigram-based BLEU scores. This shows that with some more optimization, performance could be at par with the current best models that employ self-attention to encode images and text. The lower performance in the other metrics especially the CIDEr and SPICE scores correlates with the caption results which show that while the fully connected models are able to pick up on the individual salient information available in the images or text, they are not as capable as intra-attention networks in maintaining long distance dependency and relevancy information. As a result, while the captions generated are accurate, they are noticeably shorter, which leaves out some of the details. To compensate for the reduction in the amount of context about the relationships between each caption token and the other tokens, we implement a deeper neural network to capture more of the "intra-caption" relevancy semantics. The results obtained by FC4, especially

the SPICE scores show that this allowed much better captions to be generated as more information was learned by the layers.

There are a number of other experiments that we carried out where we kept image sequence length constant while varying the caption sequence length and vice-versa. These experiments mostly showed very similar memory consumption, usually with the self-attention variant only slightly better. This is particularly surprising observation given that the fully connected feed-forward variant has about 70% more parameters and the saved models 70% larger than the self-attention variant. Part of the surprisingly good performance of the feed-forward model could be because it is quite similar to a convolutional neural network. A feed-forward neural network can be seen as a special type of CNN with a single channel and full receptive field. The similar performance of the two models in terms of memory usage in spite of the feed-forward one having a lot more parameters implies that the self-attention model actually has a worse performance per parameter. This means that if the feed-forward algorithm is refined to decompose into fewer parameters, its memory consumption can be significantly less than that of the self-attention mechanism.

From the experiments, we make some recommendations and guidelines for dealing with fully connected layers in vision-language settings, specifically image captioning.

(i) Generally, more layers are better, up to a certain point. At some point, the trade-off will not be worth it. There will be diminishing returns whereby the time and computational resources required to train the models are so high for a meager improvement in performance. This will especially be apparent during CIDEr-D optimization, whereby you may be forced to significantly reduce batch sizes leading to an explosion in the training time required.
(ii) If implementing a meshed architecture as in [25] you could benefit from meshing just a few of the top layers, meshing all layers may lead to degradation in model performance.
(iii) The GPU has a sweet spot where the batch size saturates the cores and your model can benefit the most from using the GPU. Initial small experiments should be carried out to find the ideal batch size.

6. Conclusions

We presented simple yet effective transformer-based image captioning models and performed detailed analyses of the use of feed-forward layers to both encode the images and text captions. Whereas we did not outperform the current state-of-the-art models, the results obtained where only marginally lower than those obtained in fully attentive self-attention-based image captioning models. We demonstrated that most of the transformer self-attention layers could be replaced by fully connected feed-forward layers. In this setup, the self-attention mechanism is really only required in the cross-attention sub-module. While not explicitly stated, our experiments and results also provide an insight into the significance of attention mechanisms in transformer models. The captions produced were mostly as good as those produced by their self-attention counterparts, but did exhibit some problems when it came to capturing the long-distance intra-sequence relations between the caption tokens and the detected features. We also showed that in spite of being a larger model with many more parameters than similar self-attention based models the feed-forward models used only slightly more memory at lower batch sizes and less memory at higher batch sizes that we tested. We demonstrated that there is need to revisit some of the earlier algorithms, because they could provide alternative perspectives with which to address current issues. This could lead to alternative research directions that would greatly benefit from current hardware. As part of our future work, we plan to develop much simpler models that can compete evenly with current more complex state-of-the-art models. This would make deep learning research more easily available to researchers who have no access to high-end hardware.

Author Contributions: Conceptualization, R.I.O.; methodology, R.I.O.; software, R.I.O.; validation, R.I.O., Z.Y. and J.L.; formal analysis, R.I.O., Z.Y. and J.L.; investigation, R.I.O.; resources, R.I.O. and J.L.; data curation, R.I.O.; writing—original draft preparation, R.I.O.; writing—review and editing, R.I.O., Z.Y. and J.L.; visualization, R.I.O.; supervision, J.L.; project administration, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant U2003208, in part by the Science and Technology Plan of Hunan under Grant No. 2016TP1003, and in part by the Key Technology R&D Program of Hunan Province under Grant No. 2018GK2052.

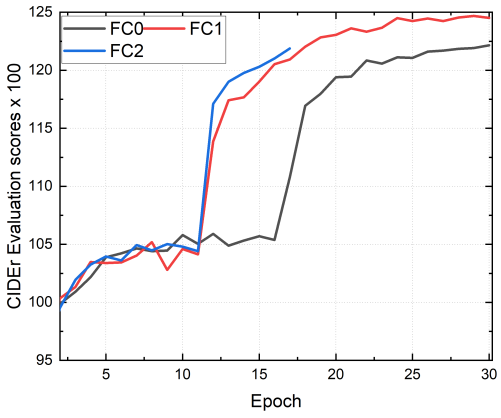
Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

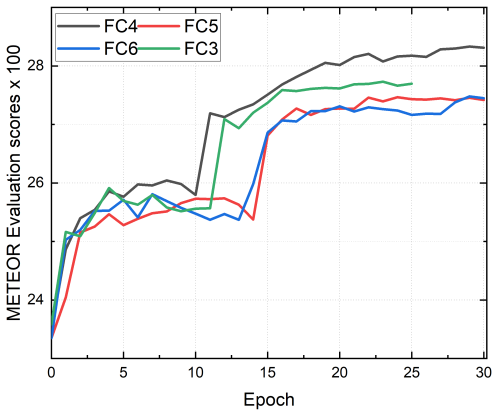
Appendix A.1. Additional Comparison Graphs

The graph in Figure A1a shows a comparison of the variants that had six or twelve layers compared to the three-layer best model, i.e., FC4. The benefits of having more layers are clearly visible including the fact that these benefits diminish the more layers we stack on top of each other.

We use the METEOR scores in Figure A1b to highlight the superior performance of the models that directly used the image features, reduced to the model dimension and forwarded to the position wise feed-forward network compared to those where it was attempted, unsuccessfully, to get a better representation of the input image features. Figure A1c focuses on the correlation between the BLEU-1 and the SPICE scores. The model was largely capable of extracting individual associations between image and region features, which resulted in decent BLEU-1 scores, but had a few issues with long-term dependencies and thus the lower SPICE and BLEU-4 metrics as further highlighted in Figure A1d. It can be observed that the poorly performing model based on the SPICE scores actually has a good performance in the BLEU-1 scores.

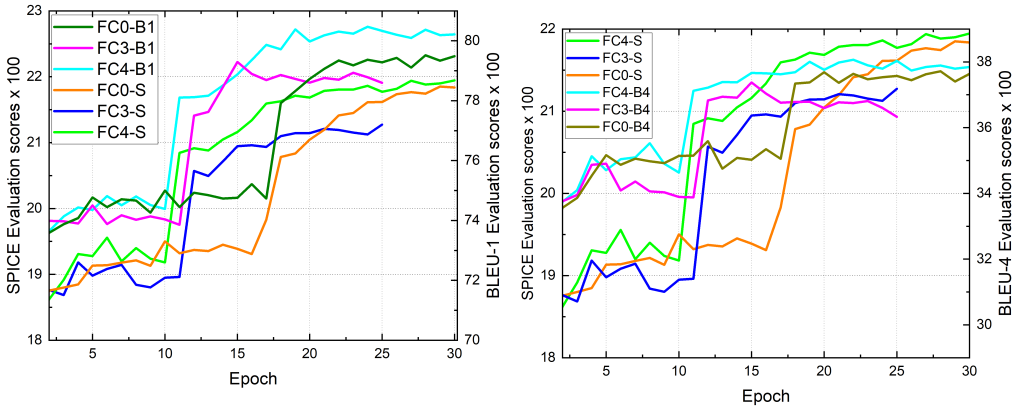


(**a**). 3 (FC0), 6 (FC1) and 12 (FC2) layer stacked variants compared.



(**b**). METEOR scores of the best model compared against the deep encoder models.

Figure A1. *Cont.*



(**c**). BLEU-1 vs. SPICE scores for select models.

(**d**). BLEU-4 vs. SPICE.

Figure A1. Graphs highlighting a comparison between specific metrics. The model name suffixes "-S", "-B1", and "-B4" denote the SPICE, BLEU-1 and BLEU-4 curve lines respectively. (**a**) Shows a comparison between the 3-, 6- or 12-layer variants to highlight the effect of stacking more layers. (**b**) Shows a comparison between the METEOR scores of the best model compared against select deep encoder models, (**c**) Shows a comparison between the SPICE and the BLEU-1 metrics for select models., and (**d**) Shows a comparison between the BLEU-4 and the SPICE metrics for select models.

Appendix A.2. Sample Captions

Tables A1 and A2 show captions from selected models. Among the captions are the hits, misses, near misses and at times, total hallucinations. There have been a lot of improvements in the quality of the captions generated, but even the state-of-the-art models show that they are nowhere near production-ready.

Table A1. Sample captions generated by the models. GT = Ground Truth, FC0, FC1, FC6, FC4 are model variants as described in Section 4.3.

	**GT**: a couple of young women sitting on the ground next to each other. **FC0**: two women sitting on the grass looking at her phone. **FC1**: two women sitting on a blanket looking at their cell phones. **FC6**: two women sitting on a blanket on her cell phones. **FC4**: two women sitting on the grass looking at their cell phones.
	**GT**: a large semi truck pulling a blue tractor behind it. **FC0**: a large truck is driving down a traffic light. **FC1**: a large dump truck driving down a traffic light. **FC6**: a construction truck carrying a crane on the road. **FC4**: a construction truck driving down a traffic light on the road.

Table A1. *Cont.*

	**GT**: group of adults and children at beach playing with various types of surfboards. **FC0**: a group of people holding surfboards on the beach. **FC1**: a group of people carrying surfboards on the beach with. **FC6**: a group of people standing on the beach. **FC4**: a group of people standing on the beach with surfboards.
	**GT**: lit up night traffic is zooming by a clock tower. **FC0**: the big ben clock tower towering over the city of london. **FC1**: the big ben clock tower towering over the street at night. **FC6**: the big ben clock tower towering over the city of london. **FC4**: the big ben clock tower towering over the city of london.
	**GT**: three girls walking in front of a parking meter. **FC0**: two women walking past a sidewalk next to a parking meter. **FC1**: two women standing next to a parking meter on the. **FC6**: two women with red hair standing next to each other. **FC4**: three women standing next to a parking meter on the street.
	**GT**: a group of kids that are standing in the grass. **FC0**: a group of people playing baseball in a field. **FC1**: a group of people playing baseball in a baseball field. **FC6**: a group of people playing baseball in a field. **FC4**: a group of children playing baseball on a baseball field.

Table A2. Sample captions generated by the models.

	**GT**: people standing near a train that is blowing smoke. **FC0**: a man standing next to a train on the tracks. **FC1**: two people standing next to a black train on the. **FC6**: a man standing next to a train on the tracks. **FC4**: two people standing next to a black train on the tracks.

Table A2. *Cont.*

	**GT**: two cute girls with a scooter and tennis raquet. **FC0**: two little girls standing between two tennis racket. **FC1**: two young girls holding tennis rackets on a tennis court. **FC6**: a little girl standing next to a tennis racket. **FC4**: two little girls holding a tennis racket on a tennis court.
	**GT**: two men riding snowboards in a snow storm down a slope. **FC0**: two snowboarders are snowboarding down a snow covered slope. **FC1**: two people are snowboarding down a snow covered slope. **FC6**: a couple of men riding snowboards in the snow. **FC4**: two snowboarders are on snowboards in the snow covered slope.
	**GT**: an old car outside near a harbor of some sort. **FC0**: a yellow car parked in front of a gas station. **FC1**: a yellow truck parked in front of a fire hydrant. **FC6**: a yellow car parked in front of a building. **FC4**: a yellow truck parked in front of a yellow fire hydrant.
	**GT**: a row of girls holding umbrellas pose for a picture. **FC0**: a group of women walking down a street with umbrellas. **FC1**: a group of women walking down the street with umbrellas. **FC6**: a group of women holding umbrellas on the street. **FC4**: a group of women walking down a street holding umbrellas.
	**GT**: a man in front of a christmas tree with his dog. **FC0**: a man wearing a christmas tree with a christmas tree. **FC1**: a dog wearing a santa hat standing in the. **FC6**: a white dog standing in front of a christmas tree. **FC4**: a man wearing a santa hat standing in front of a christmas tree.

References

1. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Technical Report; California University San Diego La Jolla Institute for Cognitive Science: San Diego, CA, USA, 1985.
2. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
3. LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1989**, *1*, 541–551. [CrossRef]

4. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008.
5. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
6. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
7. Lee-Thorp, J.; Ainslie, J.; Eckstein, I.; Ontanon, S. FNet: Mixing Tokens with Fourier Transforms. *arXiv* **2021**, arXiv:2105.03824.
8. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3156–3164.
9. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 30, pp. 2048–2057.
10. Biten, A.F.; Gomez, L.; Rusinol, M.; Karatzas, D. Good News, Everyone! Context driven entity-aware captioning for news images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 12466–12475.
11. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. 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, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6077–6086.
12. Karpathy, A.; Fei, L. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3128–3137.
13. Karpathy, A.; Joulin, A.; Fei-Fei, L.F. Deep fragment embeddings for bidirectional image sentence mapping. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 1889–1897.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
15. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
16. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
17. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
18. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
19. Wang, Q.; Chan, A.B. Cnn+ cnn: Convolutional decoders for image captioning. *arXiv* **2018**, arXiv:1805.09019.
20. Aneja, J.; Deshpande, A.; Schwing, A.G. Convolutional image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5561–5570.
21. Huang, L.; Wang, W.; Chen, J.; Wei, X.Y. Attention on attention for image captioning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019; pp. 4634–4643.
22. Li, X.; Yin, X.; Li, C.; Zhang, P.; Hu, X.; Zhang, L.; Wang, L.; Hu, H.; Dong, L.; Wei, F.; et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*; Springer: New York, NY, USA, 2020; pp. 121–137.
23. Zhou, L.; Palangi, H.; Zhang, L.; Hu, H.; Corso, J.; Gao, J. Unified vision-language pre-training for image captioning and vqa. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 13041–13049.
24. Liu, W.; Chen, S.; Guo, L.; Zhu, X.; Liu, J. Cptr: Full transformer network for image captioning. *arXiv* **2021**, arXiv:2101.10804.
25. Cornia, M.; Stefanini, M.; Baraldi, L.; Cucchiara, R. Meshed-Memory Transformer for Image Captioning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10578–10587.
26. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16 $\times$ 16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
27. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497.
28. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
29. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
30. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv* **2015**, arXiv:1508.04025.
31. Rush, A.M.; Chopra, S.; Weston, J. A neural attention model for abstractive sentence summarization. *arXiv* **2015**, arXiv:1509.00685.
32. See, A.; Liu, P.J.; Manning, C.D. Get to the point: Summarization with pointer-generator networks. *arXiv* **2017**, arXiv:1704.04368.
33. Osolo, R.I.; Yang, Z.; Long, J. An Attentive Fourier-Augmented Image-Captioning Transformer. *Appl. Sci.* **2021**, *11*, 8354. [CrossRef]

34. Lu, J.; Batra, D.; Parikh, D.; Lee, S. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv* **2019**, arXiv:1908.02265.
35. Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C.L.; Parikh, D. Vqa: Visual question answering. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2425–2433.
36. Yu, Z.; Yu, J.; Cui, Y.; Tao, D.; Tian, Q. Deep modular co-attention networks for visual question answering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 6281–6290.
37. Lu, J.; Yang, J.; Batra, D.; Parikh, D. Hierarchical question-image co-attention for visual question answering. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 289–297.
38. Zheng, Z.; Wang, W.; Qi, S.; Zhu, S.C. Reasoning visual dialogs with structural and partial observations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 6669–6678.
39. Wang, H.; Wang, W.; Shu, T.; Liang, W.; Shen, J. Active visual information gathering for vision-language navigation. In *European Conference on Computer Vision*; Springer: New York, NY, USA, 2020; pp. 307–322.
40. Brownlee, J. *Deep Learning for Natural Language Processing: Develop Deep Learning Models for Your Natural Language Problems*; Machine Learning Mastery: San Juan, Puerto Rico 2017.
41. Rennie, S.J.; Marcheret, E.; Mroueh, Y.; Ross, J.; Goel, V. Self-critical sequence training for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7008–7024.
42. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*; Springer: New York, NY, USA, 2014; pp. 740–755.
43. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318.
44. Vedantam, R.; Lawrence Zitnick, C.; Parikh, D. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4566–4575.
45. Lin, C.Y. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*; Association for Computational Linguistics: Barcelona, Spain, 2004; pp. 74–81.
46. Banerjee, S.; Lavie, A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the Acl Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, MI, USA, 29 June 2005; pp. 65–72.
47. Anderson, P.; Fernando, B.; Johnson, M.; Gould, S. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*; Springer: New York, NY, USA, 2016; pp. 382–398.
48. Jiang, W.; Ma, L.; Jiang, Y.G.; Liu, W.; Zhang, T. Recurrent fusion network for image captioning. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 499–515.
49. Bapna, A.; Chen, M.X.; Firat, O.; Cao, Y.; Wu, Y. Training deeper neural machine translation models with transparent attention. *arXiv* **2018**, arXiv:1808.07561.