Camera2Caption: A Real-Time Image Caption Generator

Pranay Mathur*, Aman Gill†, Aayush Yadav‡, Anurag Mishra§ and Nand Kumar Bansode¶
Department of Computer Engineering
Army Institute of Technology, Pune, India 411015
*Email: {*pranay360,†amangill1094,‡aayushyadav96,§anuragmishracse}@gmail.com*
and ¶nkbansode@aitpune.edu.in

***Abstract*—The recent advances in Deep Learning based Machine Translation and Computer Vision have led to excellent Image Captioning models using advanced techniques like Deep Reinforcement Learning. While these models are very accurate, these often rely on the use of expensive computation hardware making it difficult to apply these models in real-time scenarios, where their actual applications can be realised. In this paper, we carefully follow some of the core concepts of Image Captioning and its common approaches and present our simplistic encoder and decoder based implementation with significant modifications and optimizations which enable us to run these models on low-end hardware of hand-held devices. We also compare our results evaluated using various metrics with state-of-the-art models and analyze why and where our model trained on MSCOCO dataset lacks due to the trade-off between computation speed and quality. Using the state-of-the-art TensorFlow framework by Google, we also implement a first of its kind Android application to demonstrate the real-time applicability and optimizations of our approach.**

1. Introduction

For a machine to be able to automatically describe objects in an image along with their relationships or the actions being performed using a learnt language model is a challenging task, but with massive impact in many areas. For example it could help people with visually impairment better understand visual inputs, thereby acting as an assistant or a guide.

Not only must the model be able to solve the computer vision challenges of identifying the objects in an image, but it must also be intelligent enough to capture and express the object's relationships in natural language. For this reason, image caption generation has long been considered as a difficult problem. Its purpose is to mimic the human ability to comprehend and process huge amounts of visual information into a descriptive language, making it an attractive problem in the field of AI.

Many major tech-companies are investing heavily in Deep Learning and AI research, as a result of which the particular problem of image captioning is being studied at several organisations by several different teams. The two main bodies of work that form the basis of this paper are Show and Tell by Oriol Vinyal et al (2015) [1] and the more advanced, attention based Show, Attend and Tell by Kelvin Xu et al (2015) [2].

Image captioning can be used for a variety of use cases such as assisting the blind using text to speech by realtime responses about the surrounding environment through a camera feed, enhancing social medial experience by converting captions for images in social feed as well as messages to speech. Assisting young children in recognizing objects as well as learning the English language. Captions for every image on the internet can lead to faster and descriptively accurate image searches and indexing. In robotics, the perception of environment for an agent can be given a context through natural language representation of environment through the captions for the images in the camera feed.

2. Related Work

In this section we take a look at some of the work previously undertaken in this problem domain. Earlier image captioning methods relied on templates instead of a probabilistic generative model for generating the caption in natural language. Farhadi et al. (2010) [3], use triplets of `<object, action, scene>` along with the pre-determined template to generate caption. They discriminately train a multi-label Markov Random Field to predict the values of the triplets. Kulkarni et al. (2011) [4], detect objects from the image, predict a set of attributes and prepositions (spatial information against other objects) for every object, construct a labelled Conditional Random Field graph and generate sentences using the labels and a template. These approaches do not generalize well as they fail to describe the previously unseen composition of objects even if the individual objects were present in the training data. Also, the problem with template approach is their proper evaluation.

To address these issues, deep neural network architectures were used along with a language model, which co-embeds the images and captions in the same vector space. The basic approach is essentially the same – a Convolutional Neural Network (encoder) that generates a sequence of features which are fed into a sequence-to-sequence model (decoder) that learns a language model to generate natural language descriptions. Kiros et al. (2013) [5] use a convolutional neural network and a Log-Bilinear language model,

978-1-5090-5595-1/17/$31.00 ©2017 IEEE

which is a feedforward neural network with single linear hidden layer, to predict the next word given the image and previous words.

Our model is inspired by Show and Tell [1] which uses GoogLeNet CNN to extract image features and generate captions using Long Short Term Memory cells. We differ from their implementation to optimize for real-time scenarios. Show, Attend and Tell [2] makes use of new developments in machine translation and object detection to introduce an attention based model that takes into account several "spots" on the image while generating the captions. They extract features from lower convolutional layer instead of extracting from the penultimate layers, resulting in a feature vector of length $14 \times 14 \times 512$ for every image. This number can be interpreted as 196 "spots" on the image, each having a feature vector of length 512. These 196 spots are taken into consideration while generating the captions. Using visual attention, the model was able to learn to fix its sight on the important objects in the image when generating captions. They introduced two attention mechanisms, a "soft" deterministic attention mechanism trained with back-propagation methods; and a "hard" stochastic attention mechanism trained by maximizing an approximate variational lower bound. Adding attention improves the quality of the generated captions, but with a huge cost of additional trainable weights. Also, processing and storing encoded features takes a lot of computational time, thereby rendering this technique futile for real-time applications on consumer cell phone hardware. Karpathy and Li (2015) [6] introduced an approach that leveraged available datasets of images with their sentence descriptions to learn the inter-modal relationship between language and visual data. This alignment model is characterised by combining Convolutional Neural Networks on visual data regions, bidirectional Recurrent Neural Networks over language, and a structured objective aligning these modalities. The Multimodal Recurrent Neural Network architecture, thus created, uses the learnt alignments to obtain inference about the required image captions. The alignment model produces state of the art results in retrieval experiments on *Flickr8K*, *Flickr30K* and *MSCOCO* [7] datasets. Inspired by recent developments in reinforcement learning for training of deep end-to-end systems, Etienne et al (2016) [8] introduced a new optimization approach, Self-critical Sequence Training for image captioning, which is a modified form of the REINFORCE algorithm [9]. Their approach addresses the issue of exposure bias [15] problem and they optimize their model directly for non-differentiable evaluation metrics like BLEU, ROUGE, etc., instead of a cross-entropy loss.

3. Architecture

3.1. Model

In our implementation we follow an approach similar to Show and Tell [1] by introducing an encoder-decoder architectural system. The encoder being the pre-trained InceptionV4 Convolutional Neural Network by Google [16] and

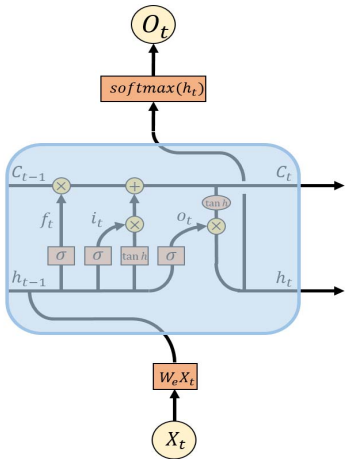

Figure 1. LSTM Cell at timestep t with 3 gated layers – forget gate layer f_t, input gate layer i_t, output gate layer o_t and – state tuple C_t the memory cell state and h_t the hidden state. Vector operations are represented by circles and learnable neural network layers with rectangles. Merging lines denote concatenation of two vectors, while lines splitting denote copies of the same vector.

the decoder, a Deep Recurrent Neural Network with Long Short Term Memory Cells. Encoder InceptionV4 is used to transform raw images I into a fixed length embedding F which represent the convolved features for the images. These embeddings are obtained by running a forward pass till the penultimate layer i.e., the *average pool layer* of the InceptionV4 model.

The decoder in our model has two phases, namely, training and inference. The decoder is responsible for learning the word sequences given the convolved features and original caption. The decoder's hidden state h_t is initialised using these image embeddings features F at timestep $t = 0$. Hence the basic idea of encoder-decoder model is demonstrated by the following equations.

$$F = encoder(I); X_{t=0} = F; O_t = decoder(X_{t:0 \to t})$$

The training process in the RNN with LSTM Cell based decoder works on a probabilistic model in which the decoder maximizes the probability of word p in a caption given the convolved image features F and previous words $X_{t:0 \to t}$. To learn the whole sentence of length N corresponding to the features F the decoder uses its recurrent nature to loop over itself over a fixed number of timesteps N with the previous information (features and sampled words at timestep t) stored in its cell's memory as a state. The decoder can alter the memory C_t as it unrolls by adding new state, updating or forgetting previous states through the LSTM's forget f_t, input i_t and output o_t memory gates.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C_t} = \sigma(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C_t} \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

$$O_t = argmax(softmax(h_t)) \quad (7)$$

The transformation of the the t^{th} input x_t at timestep t into an t^{th} output word O_t are guided by the above mentioned equations. Where (W_f, b_f), (W_i, b_i) and (W_o, b_o) are learnable weight vectors and bias vectors in the model activated by σ sigmoid and $tanh$ hyperbolic tangent non linearities.

Here, each word X_t is transformed into fixed length vectors $W_e X_t$ using a Word Embedding W_e of dimension $V \times W$ where V is the no. of words in vocabulary and W is the length of unqiue embedding for every word in vocabulary. These representations are learnt during the training process through a Word2Vec model.

Thus, the final objective Z of the decoder training process is to maximize the probability p of occurrence of a word at a timestep t given cell state C_t, hidden state h_t, features F (represented jointly as ϕ_t), previous ground truth words $|X_{t:0\rightarrow t-1}$ over the pre-defined model hyperparameters β discussed in Implementation section. Objective Z is maximized by minimizing the loss function L that we use is the summation of the negative log likelihood of the correct word at each timestep, this can also be referred as cross entropy of the probability distribution of word sampled at each timestep.

$$Z = \arg\max_{\beta}(\sum_{t=0}^{N} \log(p(O_t|X_{t:0\rightarrow t-1}, \phi_t; \beta)))$$

$$L = H(u, v) = \text{minimize}(\sum_{t=0}^{N} -u(X_t)\log(v(O_t)))$$

where $H(u, v)$ is the cross-entropy. u and v are softmax probability distributions of ground truth word X_t and generated word O_t at timestep t.

During inference the input image is forward passed through the encoder to obtain the convolved features which are subsequently passed through the decoder. At timestep $t = 0$ the decoder samples start token $O_{t=0}$ =<S> given the input features F and for the subsequent timesteps the decoder samples a new word based on input features as well as previously sampled words $O_{t:0\rightarrow t}$ until an end token </S> is encountered at timestep $t = N$.

3.2. Application

In our implementation we customize and optimize our encoder-decoder model to function in a real-time environment and to work on mobile devices. For this we use Google's numerical computation library, TensorFlow. The main advantage of using TensorFlow for any Deep Learning model is its data-flow graphs based computations, which basically comprise of nodes, representing mathematical operations and edges, representing the multidimensional data arrays (tensors) communicated between nodes. The flexible architecture of TensorFlow enabled us to deploy our model's computation on CPUs and GPUs and thus helped us utilize inherent parallelism in primitive operations and computations.

Figure 3. **Application screenshots**

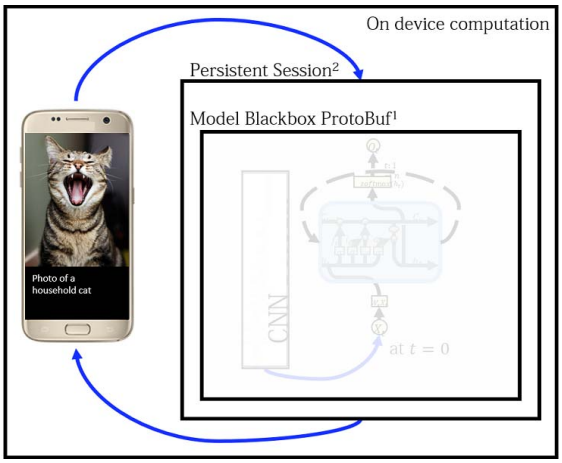

Figure 4. **Application Architecture**

Apart from the encoder-decoder training and inference, we also implement a graph-based pre-processing method for re-shaping the input image to be compatible with our model, the graph based approach gives a 6x speed up in pre-processing image before inference.

During the training process of the model using TensorFlow, files related to checkpoints and computation graph's definition and metadata are generated after every training epoch. The checkpoint file contains trained weights and the graph definition file contains information related to associating these weights in a meaningful manner to re-create and re-use the model for inference as well as training.

We combine these files for pre-processing module, the encoder and the decoder to form three protocol buffer [1] files. These ProtoBuf files allowed us to make three separate

1. Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data, hence can store computation graph as well as trained weights

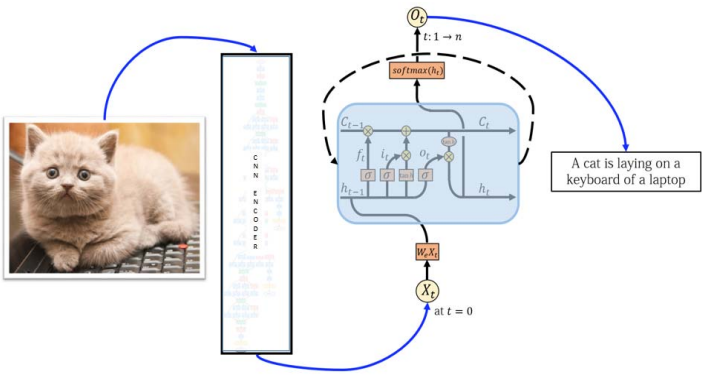

Figure 2. **Model Architecture**

abstract modules. Further, we stitched these three modules to make an abstract end-to-end model, hence simplifying the deployment to serve static as well as real-time requirements. We prepare a final ProtoBuf file which acts as an Image Captioning Black Box to generate caption for an input image and, at the same time, simplifying usability and deployability as well as abstracting unnecessary complexities (such as separate reshaping of image, feature extraction and forward passes) from the end-user or system. These techniques enabled us to gain significant speed up in real-time caption generation, a forward pass using a single session [2] through our Black Box model was much faster than creating three separate session for above mention three modules and forward passing through them individually. Thus, a single session runs persistently throughout the life cycle of the application which serves appropriate captions to real-time input images from the camera feed.

4. Experiments

4.1. Dataset

Initially, our model was trained on *Flickr30k* dataset with 31783 images with five captions each, but owing to less number of training samples and every training caption beginning with "A man ... " template, our model failed to generalize, so we shifted to *MSCOCO* (2014) [7] training dataset with 82780 images, each with five ground truth captions. For offline evaluation, we use the karpathy split[3]. Although, this split of 5000 images is not a standardized split, but it has been used by many researchers to report their results.

4.2. Implementation

With optimizations for quicker caption generation as our objective, our model, which is inspired by Vinyals et al (2015) [1] differs from their implementation in the following ways:

- Our encoder, InceptionV4, which uses residual connections, not only performs better than GoogLeNet used by Vinyals on the ImageNet task, but is also faster.
- The LSTM's hidden dimension, word and image embeddings in our model are all fixed to 256, instead of 512.
- In our model, the encoder and decoder are stitched into a single graph, hence only a single TensorFlow session needs to be loaded to run the entire model.
- We generate captions using a single trained model, instead of using an ensemble of trained models.
- To avoid the overhead of exploring the complete search space of beam search tree, we generate captions greedily, hence speeding up the real-time inference.

During training, we use the average pooling layer (after the final convolutional layer) from a pre-trained inceptionV4 to encode the image (resized to $299 \times 299 \times 3$) resulting in a vector of dimension 1536. Currently our model supports only JPEG and PNG image formats. The rationale behind using a pre-trained encoder instead of training one was to avoid over-fitting. We then pre-process the captions by lower casing them, replacing words with frequency of occurring in the training dataset less than or equal to two, by "UNK", truncating the caption's length to twenty words and prepending and appending the caption with start (<S>) and stop (</S>) tokens. Our vocabulary's size is 14383. The LSTM's hidden dimension, word and image embeddings are all fixed to 256. The weights of the decoder and the embeddings were randomly initialized. We use cross-entropy loss function with ADAM optimizer [10] for training the model, with initial learning rate as 2×10^{-3}. The learning rate decays as:

$$decayed\ rate = learning\ rate \times decay\ rate^{\frac{global\ step}{decay\ steps}}$$

We fix the $decay\ rate$ and $decay\ steps$ to 0.95 and 100000. The model weights are saved as checkpoints after every

2. Session is a TensorFlow object to run through a defined Computation Graph

3. http://cs.stanford.edu/people/karpathy/deepimagesent/

TABLE 1. EVALUATION OF DIFFERENT MODELS AGAINST BLEU-4, ROUGE-L, METEOR AND CIDER METRICS, α INDICATES AN ENSEMBLE, ρ INDICATES A DIFFERENT SPLIT, – INDICATES THAT THE VALUE WASN'T AVAILABLE.

Model	B-4	R-L	METEOR	CIDEr
$NIC^{\alpha\rho}$ [1]	27.7	–	23.7	85.5
$NICv2^{\alpha\rho}$ [17]	32.1	–	25.7	99.8
$SCSTw/oAttention^{\alpha}$ [8]	33.4	55.4	26.2	110.4
$SCSTw/Attention^{\alpha}$ [8]	34.8	56.3	26.9	115.2
Xu (Hard Attention) [2]	25.0	–	23.04	–
Karpathy [6]	23.0	–	19.5	66.0
Our Model	22.7	48.0	22.0	74.7
$Random^{\rho}$	4.6	–	9.0	5.1
$Human^{\rho}$	21.7	48.4	25.2	85.4

TABLE 2. EVALUATION OF DIFFERENT MODELS AGAINST BLEU-1, BLEU-2, BLEU-3 AND BLEU-4.

Model	B-1	B-2	B-3	B-4
Xu (Hard Attention) [2]	71.8	50.4	35.7	25.0
Xu (Soft Attention) [2]	70.7	49.2	34.4	24.3
Karpathy [6]	62.5	45.0	32.1	23.0
Our Model	65.0	46.9	36.2	22.7

epoch, making our model re-trainable. During testing, the generated caption's length is truncated to 20 words. Captions are generated greedily instead of using beam search to reduce the time for caption generation.

4.3. Results

We evaluate our generated captions using BLEU [11], ROUGE [14], METEOR [13] and CIDEr [12] scores computed with the coco-caption code provided as a part of the COCO evaluation toolkit[4]. We report our results in Table 1 and Table 2.

We focus our implementation for quicker caption generation instead of achieving state-of-the-art performance and due to this there exist a few challenges for a justified comparison. The first challenge is that we train and perform inference using only a single model, whereas some of the other implementations use an ensemble to boost their score. The second challenge is that our model doesn't use visual attention, which adds to the overall scores but at a high cost of additional parameters, making the inference process slower. The third challenge is the differences of dataset splits, due to the lack of a standardized split for offline evaluation. The final challenge is that we generate captions greedily, instead of using beam search and exploring the search space which improves the performance but consumes more time.

4. https://github.com/tylin/coco-caption

5. Conclusion

We create and deploy a first of its kind mobile device based application to introduce possible use-cases for our model. We also test our model against other recent works in image captioning. By making use of Inception architecture and by simplifying the overall flow design, we optimize our model to perform well in real time (on mobile devices) and manage to obtain high quality captions.

References

[1] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2015) Show and tell: A neural image caption generator. *CVPR 1, 2*

[2] K. Xu (2015) Show, attend and tell: Neural image caption generation with visual attention. *inProc. Int. Conf. Mach. Learn.*

[3] Farhadi A. et al. (2010). Every Picture Tells a Story: Generating Sentences from Images. *Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision – ECCV 2010. Lecture Notes in Computer Science*, vol 6314. Springer, Berlin, Heidelberg

[4] Kulkarni G, Premraj V, Dhar S, Li S, Choi Y, Berg AC, Berg TL (2011) Baby Talk: Understanding and Generating Image Descriptions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (20-25 June 2011)

[5] R. Kiros and R. Z. R. Salakhutdinov (2013) Multimodal neural language models. *inProc. Neural Inf. Process. Syst. Deep Learn. Workshop*

[6] Andrej Karpathy, Li Fei Fei (2015) Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (April 2017), vol 39, issue 4:664 – 676

[7] T.-Y. Lin, et al (2014) Microsoft COCO: Common objects in context. *arXiv:1405.0312*

[8] Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross and Vaibhava Goel (2016) Self-critical Sequence Training for Image Captioning. *in arXiv:1612.00563*

[9] Ronald J. Williams (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *In Machine Learning, pages 229–256, 1992. 2, 3*

[10] Diederik P. Kingma and Jimmy Ba (2015) Adam: A method for stochastic optimization. *ICLR, 2015. 2, 5*

[11] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu (2002) BLEU: A method for automatic evaluation of machine translation. *in Proc. 40th Annu. Meeting Assoc. Comput. Linguistics, 2002, pp. 311–318*

[12] R. Vedantam, C. L. Zitnick, and D. Parikh (2015) CIDEr: Consensus based image description evaluation. *in arXiv:1411.5726*

[13] S. Banerjee and A. Lavie (2005) Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. *in Proc. ACL Workshop Intrinsic Extrinsic Eval. Measures Mach. Transl. Summarization, vol. 29, pp. 65–72.*

[14] C.-Y. Lin (2004) Rouge: A package for automatic evaluation of summaries. *in Proc. ACL Workshop Text Summarization Branches Out, vol. 8*

[15] Marc' Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. (2015) Sequence level training with recurrent neural networks. *ICLR, 2015. 1, 2, 3, 4, 5, 10*

[16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi (2016) Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *in arXiv:1602.07261*

[17] Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan (2016) Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge *PAMI*

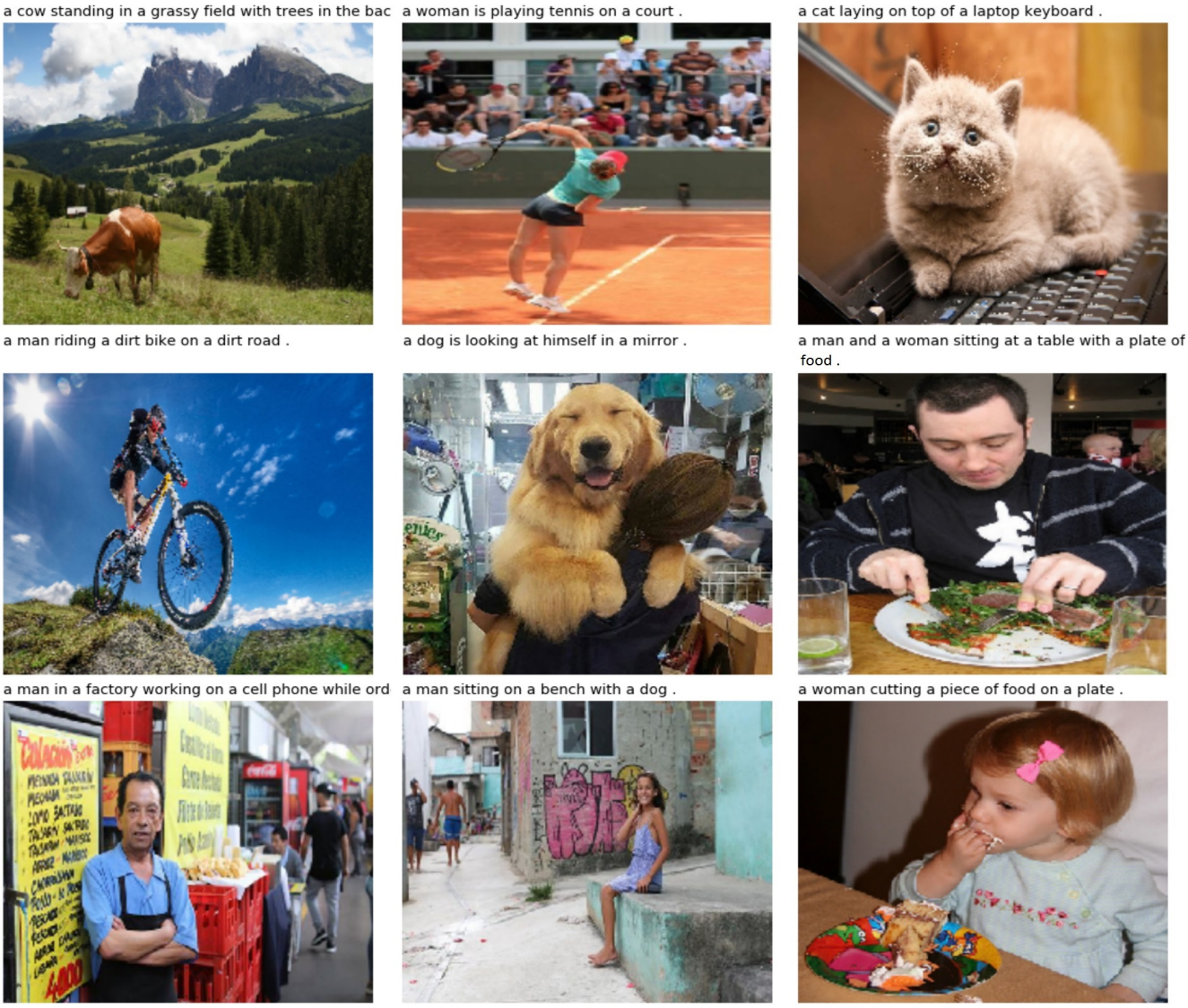

Figure 5. **first row: Correct Captions, second row: somewhat correct captions, third row: incorrect captions**