Optimizing Image Captioning through Decoder Enhancements

Ahmad Umair
aumair@berkeley.edu
UC Berkeley – MIDS – W266

Abstract

This article highlights the enhancements to the show-and-tell architecture, a traditional image captioning encoder-decoder model. Our study focused on the arrangement and quantity of long short-term memory (LSTM) units in the decoder component of the architecture. By introducing two models, merge-2x and hybrid-merge-inject, we observed respective enhancements of 17 and 19 in BLUE-3 scores on the COCO dataset.

1 Introduction

Image captioning has various vital applications that significantly improve accessibility, efficiency, and user experience in different industries. This technology provides image text descriptions, making visual content accessible to people with visual impairments. In e-commerce, image captioning automatically creates detailed product descriptions, making it easier for customers to search and interact with products. Social media platforms use this technology to provide automatic alt text for images, making content more inclusive and discoverable. In digital archiving and journalism, image captioning helps organize and retrieve visual data by creating descriptive metadata, which makes managing and accessing extensive image collections more efficient. By connecting visual and textual information, image captioning enhances our understanding of visual data, profoundly enriching our interaction with and understanding of visual data (Bai and An, 2018; Wang et al., 2020; Agarwal and Verma, 2024).

Computer vision and natural language processing (NLP) have significantly advanced image captioning, enabling machines to comprehend and articulate visual content. Computer vision techniques, particularly convolutional neural networks (CNNs), extract intricate features from images, identifying objects, actions, and scenes. NLP, employing models such as recurrent neural networks (RNNs) and transformers, generates coherent and contextually relevant captions by translating visual information into natural language. The synergy of these technologies facilitates the creation of precise and meaningful image descriptions, thereby enhancing accessibility and user experience across diverse applications.

In this article, we explored the modifications made to the frameworks outlined in the influential article on image captioning: "Show and Tell: A Neural Image Caption Generator" (Vinyals et al. 2015). The fundamental architecture still relies on CNNs and RNNs – a fixed-length CNN vector is generated by the CNN acting as an encoder, and an RNN decoder generates the caption. Our investigation focused on rearranging the RNNs at various architectural positions to enhance BLEU scores compared to the models mentioned earlier. Two models, Merge-2x and Hybrid-inject-merge, were proposed to enhance the Show and Tell architecture, resulting in BLEU-3 score increases of 17 and 19, respectively, compared to the baseline model.

2 Background

The field of image captioning has made significant strides recently, driven by the development of various models, including those employing RNNs for detailed descriptions and advanced methods with improved attention and learning capabilities. To further elevate this area, there is a strong emphasis on transformer models. Advancements in clip image generation have significantly improved the ability to create detailed and contextually accurate images from text prompts, thanks to the integration of powerful AI models like Contrastive Language-Image Pre-Training (CLIP). These models have revolutionized fields such as art, design, and content creation by enabling users to generate highly customized and imaginative visuals with minimal input (Yu et al., 2019; Mokady et al., 2021).

In the image caption generation model depicted in Figure 1, a pre-trained CNN extracts feature vectors from an input image, which are then input into an LSTM network. The LSTM employs the SoftMax activation function to classify words and generate captions by embedding them to form comprehensive sentences describing the image.

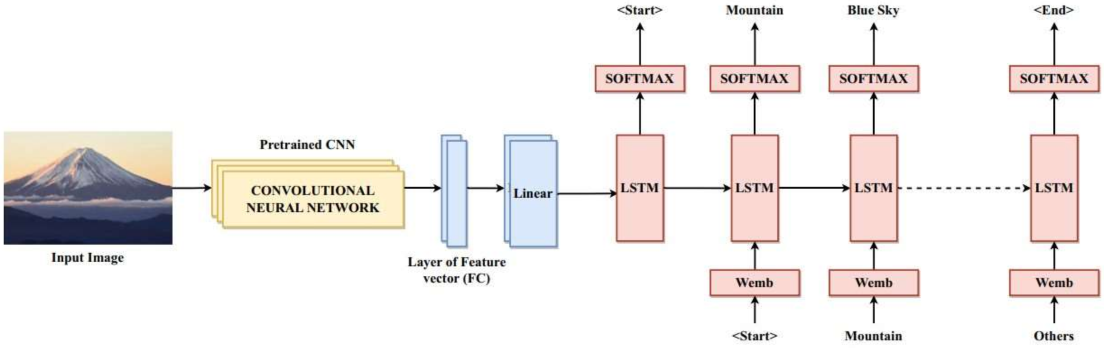


Figure 1: Image Caption Generator using CNN model (Agarwal and Verma, 2024)

Vinyals et al.'s architecture harnesses CNN with a unique batch normalization approach. They employed randomly initialized weights for all components except for the CNN weights, which were left unchanged due to the potential adverse effects of modification. Moreover, they utilized 512 dimensions for the embeddings and LSTM memory size, achieving a remarkable 27.2 BLEU-4 score on the COCO dataset (Vinyals et al., 2015).

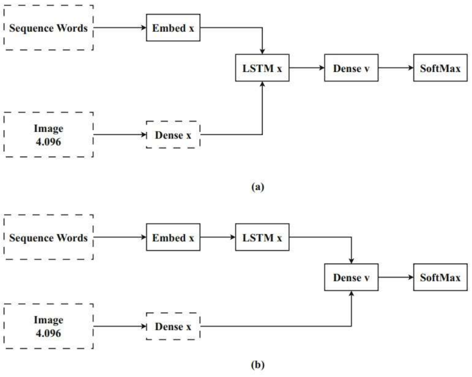


Figure 2: Two show-and-tell approaches are a) Inject architecture, and b) Merge Architecture (Tanti et al., 2018)

Tanti et al. (2018) delved into the role of recurrent neural networks (RNNs) in image captioning architecture. Two main architectures were explored. One approach, the Inject Architecture, involves integrating image features into the RNN during sequence encoding as the primary generation component. The other, called the Merge Architecture, keeps linguistic and visual features separate and merges them in a later multimodal layer for predictions.

The "Show, Attend, and Tell" paper bolsters the approach by integrating an attention mechanism. This mechanism allows the model to focus on different parts of the image while generating each word in the caption, enhancing the descriptions' relevance and detail (Xu et al., 2015).

In this work, we explored improvements in the merge architecture and integrated the merge and inject architecture.

3 Methods

3.1 Dataset

Several publicly available datasets exist for image captioning, such as Flick8k, Flickr30k, and Google conceptual captions. However, we opted to utilize the COCO (Microsoft Common Objects in Context) dataset for our research. This extensive dataset is annotated and comprises over 330,000 images featuring labeled objects across 80 categories, object segmentation masks, human pose keypoints, and captions.

The COCO dataset underwent a significant update in 2017. The 2017 dataset consists of roughly 118,000 images for training and 5,000 for validation, with five captions provided for each image. Due to memory issues and time constraints, we focused on the Animals category, which includes approximately 27,000 images for training and 800 for validation. The following subsection provides more information and reasons for using the COCO Animals category.

3.2 Image Feature Extraction – Encoder

In line with previous studies (Wang et al., 2020; Agarwal and Verma, 2024), we employed a pre-trained CNN for feature extraction instead of creating a new CNN. Specifically, we utilized the Xception model. The architecture of the Xception model is based on depthwise separable convolutions, which enhance traditional convolutions by segregating spatial and depthwise operations. This leads to enhanced performance and computational efficiency, rendering it suitable for various tasks such as image classification, object detection, and image captioning. The last classification layer was removed to get the feature vector of dimension 2048.

We faced a challenge with the size of the COCO dataset, which includes approximately 120,000 images. Extracting feature vectors for all these images using the pre-trained Xception model was estimated to take around four days. Additionally, memory issues were causing delays in the training phase. As a result, we decided to train our models on a subset of the COCO dataset, explicitly focusing on the "Animal" supercategory, which consists of around 27,000 images. This approach allowed us to proceed with our training and testing more efficiently.

3.3 Data Generator

Next, the captions were tokenized, and a data generator produced word sequences for training. Table 1 shows an example of the training sequence.

Table 1: Input I, Image input vector (2048), and Input II, word sequence and Output word, during training

Input I	Input II	Output
Img Vec	<start>	a
Img Vec	<start> a	giraffe
Img Vec	<start> a giraffe	eating
Img Vec	<start> a giraffe eating	a
Img Vec	<start> a giraffe eating a	shrub
Img Vec	<start> a giraffe eating a shrub	<end>

The following section will cover additional details about the model architecture and training.

3.4 Model Architectures

We proposed two improvements to the show-and-tell architecture and one improvement to the show-attend-and-tell architecture.

The merge architecture is regarded as the baseline model for the show-and-tell approach. Figure 3(a) shows that the initial proposed design, Merge-2x, incorporates two LSTMs. While these two LSTMs are not directly stacked, the concept is derived from stacked LSTMs, as stacking enhances learning by capturing intricate patterns and long-range dependencies through multiple layers, resulting in improved performance and better generalization in tasks such as language modeling and machine translation.

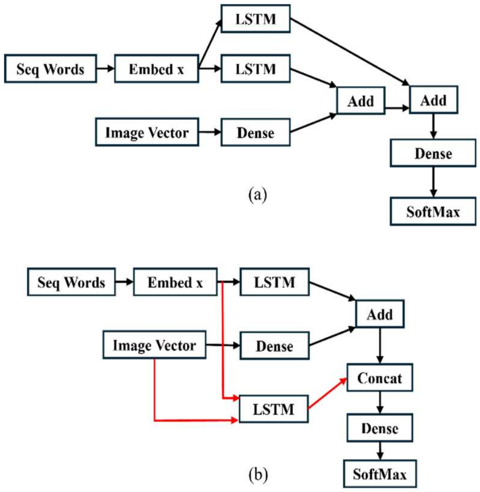


Figure 3: Proposed show-and-tell improved models, (a) Merge-2x, (b) Hybrid Merge-Inject

Figure 3(b) shows the following architecture, the hybrid model. It incorporates both merge and inject architecture and is hence named accordingly. The red edges in the model represent the injected part, while the back edges represent the merge part. The merge model in image captioning keeps visual and linguistic features separate until combined, offering flexibility and modularity in feature processing but adding complexity and potentially limiting the RNN's direct interaction with image features. In contrast, the inject model integrates image features directly into the RNN, allowing for real-time use of visual context during caption generation. This can enhance contextual understanding but may increase complexity and lead to performance issues if not managed well. Each approach has its trade-offs, with the merged model favoring separation and modularity, while the injected model emphasizes direct interaction and real-time processing. Therefore, the proposed hybrid model aims to leverage the strengths of both methods for improved performance.

3.5 Model Training

Using the data generator function mentioned in section 3.3, a batch of data for a single image was generated. During the training phase, models were trained and saved after three epochs for five iterations. These saved models were evaluated based on the metric discussed in the next section.

3.6 Model Evaluation

The Bilingual Evaluation Understudy (BLEU) scores have been utilized to assess the quality of generated captions by comparing them to reference captions within the dataset. These scores gauge the similarity between the generated and reference captions, with BLEU-n metrics evaluating n-grams while disregarding word order. The scores are on a scale from 0 to 1 (or as a percentage), where a higher score indicates a closer resemblance to the reference captions. The metric emphasizes the match of different lengths (n = 1, 2, 3, 4) and penalizes very brief captions. However, it is not as effective at capturing long-term connections and overall meaning and typically demonstrates a decline as the value of n increases (Papineni et al., 2002).

4 Results and Discussion

Before comparing the BLEU scores between our experiments, comparing our baseline show-and-tell model with the official show-and-tell results would be beneficial. In Table 2, our baseline model demonstrates a BLEU-4 score of zero, while Vinyals et al. (2015) achieved a score of 27.7. This variance can be attributed to the time constraints and computational complexities we encountered during our experimentation.

Since the baseline model has a BLUE-4 score of zero, we will focus on the BLUE-3 scores to compare the models more effectively. When comparing the Merge-2x model with the baseline model, we observed a 17% increase in the BLEU-3 score. This architecture does not use the stacked LSTM directly but instead employs them staggered. The output of the first merge, in addition to the embedded text, serves as an input to the second merge. This approach trains the weights of two LSTMs based on different outputs, and thus, it is expected to outperform a simple stacked LSTM approach.

In the Hybrid Merge-Inject mode, we observe a further 2% increase in the BLUE-3 score compared to the Merge-2x. Both the merge and inject approaches have their tradeoffs. While the merge prioritizes separation and modularity, the inject model concentrates on direct interaction and real-time processing. Combining them allows one part of the model to utilize the visual context unchanged while the other part continuously updates it. This flexibility enhances the overall performance of the model.

Several limitations constrain the show-and-tell architecture. It lacks an attention mechanism, which hinders its ability to focus on relevant parts of the image. It employs a fixed-length vector representation, potentially losing crucial details and spatial relationships. Moreover, it struggles to capture long-term dependencies, demanding high computational resources for training and potentially underperforming on complex scenes or diverse datasets. These constraints and time

Table 2: Evaluation of models based on BLEU scores.
Show-and-tell baelinse, Merge-2x and Hybrid-Merge-Inject are measured in this study.
*Show-attend-and-tell baseline is reported by Xu et. al (2015)

Model		Dataset	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Show-and-Tell	**Baseline (Merge)**	COCO Animal	33.77	5.52	0.1	0
	Merge-2x		48.73	30.85	17.32	7.54
	Hybrid Merge-Inject		51.38	34.36	19.71	8.0
Show-Attend-and-Tell	**Baseline*** Xu et al. 2015	COCO	71.85	50.4	35.7	25.0

limitations contributed to less accurate captions in our show-and-tell experiments.

5 Conclusions

In this study, we examined enhancements to the traditional show-and-tell image captioning model, focusing on the arrangement and quantity of LSTMs within the decoder component. By introducing two novel models, Merge-2x and Hybrid Merge-Inject, we achieved respective BLEU-3 score improvements of 17% and 19% on the COCO dataset, demonstrating significant advancements in generating accurate and coherent image captions.

Our findings indicate that the Hybrid Merge-Inject model, which combines the strengths of both merge and inject architectures, offers superior performance by maintaining the flexibility of feature processing while enhancing contextual understanding. Despite the improvements, the Show and Tell architecture still faces limitations, such as the lack of an attention mechanism and challenges in capturing long-term dependencies, highlighting the need for further research to address these issues and enhance image captioning capabilities.

6 References

Agarwal, L. and Verma, B., 2024. From methods to datasets: A survey on Image-Caption Generators. *Multimedia Tools and Applications*, *83*(9), pp.28077-28123.

Bai, S. and An, S., 2018. A survey on automatic image caption generation. *Neurocomputing*, *311*, pp.291-304.

Mokady, R., Hertz, A. and Bermano, A.H., 2021. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.J., 2002, July. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

Tanti, M., Gatt, A. and Camilleri, K.P., 2018. Where to put the image in an image caption generator. *Natural Language Engineering*, *24*(3), pp.467-489.

Vinyals, O., Toshev, A., Bengio, S. and Erhan, D., 2015. Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).

Wang, H., Zhang, Y. and Yu, X., 2020. An overview of image caption generation methods. *Computational intelligence and neuroscience*, *2020*(1), p.3062706.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R. and Bengio, Y., 2015, June. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048-2057). PMLR.

Yu, J., Li, J., Yu, Z. and Huang, Q., 2019. Multimodal transformer with multi-view visual representation for image captioning. *IEEE transactions on circuits and systems for video technology*, *30*(12), pp.4467-4480.

Appendices

Appendix A: COCO Dataset EDA

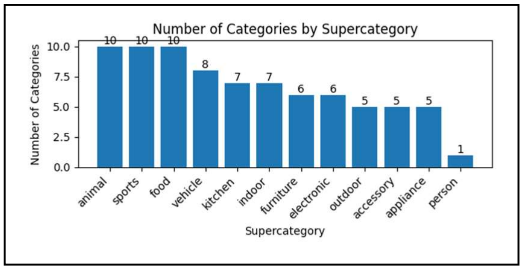


Figure 6: COCO categories by supercategories

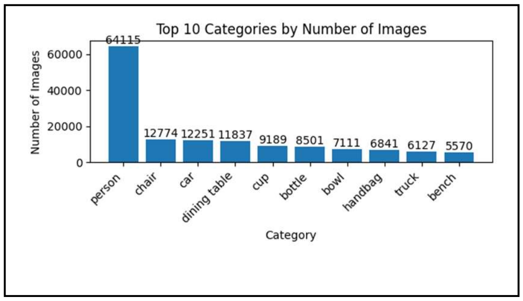


Figure 5: Top 10 categories by image count

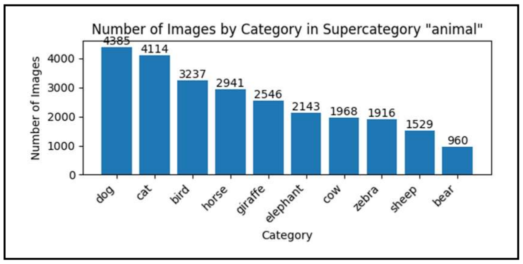
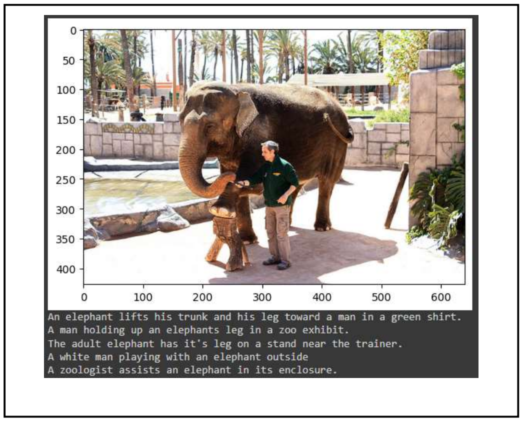


Figure 4: Image types in super category animal



```
An elephant lifts his trunk and his leg toward a man in a green shirt.
A man holding up an elephants leg in a zoo exhibit.
The adult elephant has it's leg on a stand near the trainer.
A white man playing with an elephant outside
A zoologist assists an elephant in its enclosure.
```

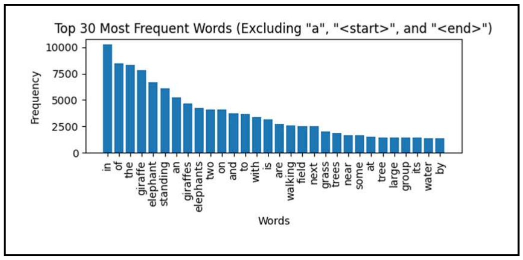
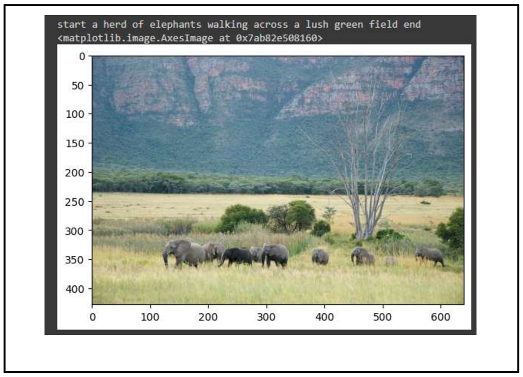


Figure 9: Words frequency in COCO animal



```
start a herd of elephants walking across a lush green field end
<matplotlib.image.AxesImage at 0x7ab82e508160>
```

Figure 10: An example of captioned image

Appendix B: Models

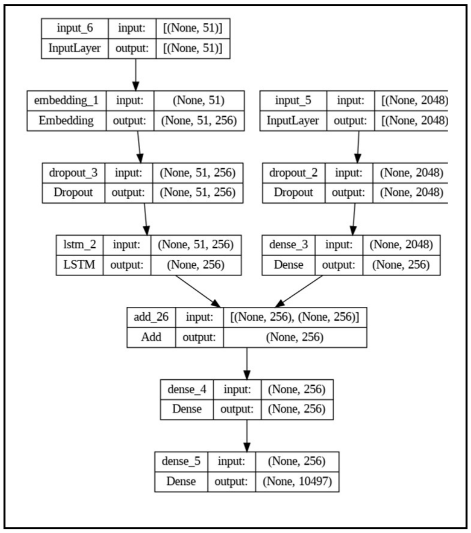

input_6 | input: | [(None, 51)]
InputLayer | output: | [(None, 51)]
embedding_1 | input: | (None, 51)
Embedding | output: | (None, 51, 256)
input_5 | input: | [(None, 2048)
InputLayer | output: | [(None, 2048)
dropout_3 | input: | (None, 51, 256)
Dropout | output: | (None, 51, 256)
dropout_2 | input: | (None, 2048)
Dropout | output: | (None, 2048)
lstm_2 | input: | (None, 51, 256)
LSTM | output: | (None, 256)
dense_3 | input: | (None, 2048)
Dense | output: | (None, 256)
add_26 | input: | [(None, 256), (None, 256)]
Add | output: | (None, 256)
dense_4 | input: | (None, 256)
Dense | output: | (None, 256)
dense_5 | input: | (None, 256)
Dense | output: | (None, 10497)

Figure 11: Baseline model (Merge)

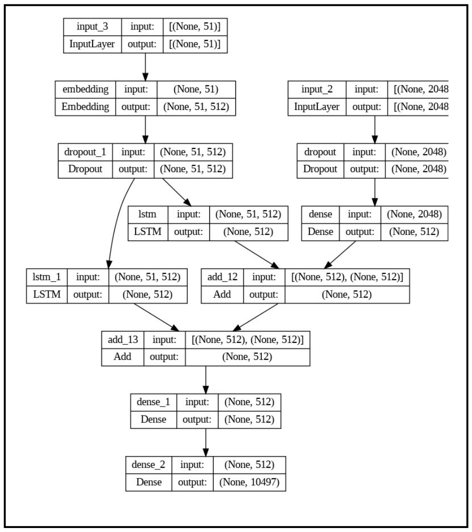

input_3 | input: | [(None, 51)]
InputLayer | output: | [(None, 51)]
embedding | input: | (None, 51)
Embedding | output: | (None, 51, 512)
input_2 | input: | [(None, 2048
InputLayer | output: | [(None, 2048
dropout_1 | input: | (None, 51, 512)
Dropout | output: | (None, 51, 512)
dropout | input: | (None, 2048)
Dropout | output: | (None, 2048)
lstm | input: | (None, 51, 512)
LSTM | output: | (None, 512)
dense | input: | (None, 2048)
Dense | output: | (None, 512)
lstm_1 | input: | (None, 51, 512)
LSTM | output: | (None, 512)
add_12 | input: | [(None, 512), (None, 512)]
Add | output: | (None, 512)
add_13 | input: | [(None, 512), (None, 512)]
Add | output: | (None, 512)
dense_1 | input: | (None, 512)
Dense | output: | (None, 512)
dense_2 | input: | (None, 512)
Dense | output: | (None, 10497)

Figure 12: Merge 2x

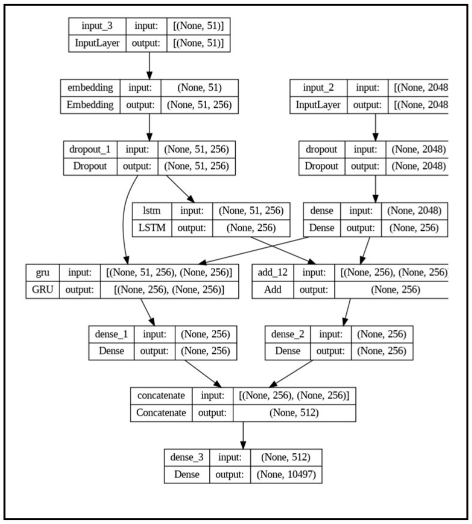

input_3 | input: | [(None, 51)]
InputLayer | output: | [(None, 51)]
embedding | input: | (None, 51)
Embedding | output: | (None, 51, 256)
input_2 | input: | [(None, 2048
InputLayer | output: | [(None, 2048
dropout_1 | input: | (None, 51, 256)
Dropout | output: | (None, 51, 256)
dropout | input: | (None, 2048)
Dropout | output: | (None, 2048)
lstm | input: | (None, 51, 256)
LSTM | output: | (None, 256)
dense | input: | (None, 2048)
Dense | output: | (None, 256)
gru | input: | [(None, 51, 256), (None, 256)]
GRU | output: | [(None, 256), (None, 256)]
add_12 | input: | [(None, 256), (None, 256)
Add | output: | (None, 256)
dense_1 | input: | (None, 256)
Dense | output: | (None, 256)
dense_2 | input: | (None, 256)
Dense | output: | (None, 256)
concatenate | input: | [(None, 256), (None, 256)]
Concatenate | output: | (None, 512)
dense_3 | input: | (None, 512)
Dense | output: | (None, 10497)

Figure 13: Hybrid-Merge-Inject