CS7643 Project Report: Exploring model architectures for image caption task on COCO

Kun Yu
kyu311@gatech.edu

Quan Qi
qqi33@gatech.edu

Runfeng Zhang
rzhang668@gatech.edu

Yuan Shang
yshang42@gatech.edu

Georgia Institute of Technology

Abstract

Image Captioning is the process of generating textual descriptions of an image. The most common deep-learning-based solutions address this task with a visualencoderNLP-decoder architecture. In this project, we explored different visual-encoder and NLPdecoder structures, and test their performances on COCO2014 datasets. To make our models comparable, in the baseline architecture (Model1), We used RESNET101 for the visual-encoder, and LSTM for the NLP-decoder with attention mechanisms. This baseline model have a bleu4=0.3372 in the evaluation dataset in our best-trained model.

For the visual-encoder module, in recent years, several non-CNN-based structures such as MLPMixer and ViT, which also show good performance in computer vision tasks. We applied MLPMixer as an alternative visual-encoder (Model2), and it showed competing performance with the baseline architecture based on its bleu4=0.2996.

For the NLPdecoder module, we first tested the performance without attention mechanisms(Model3, bleu4=0.3388). Then, we applied transformers. A transformer includes a sub-encoder and a sub-decoder part. We implemented different transformer structures, including 1. transformer with only the sub-encoder part (Model4,bleu4=0.1399); 2. transformer with only the subdecoder part (Model5,bleu4=0.3210); 3. transformer with both sub-encoder and sub-decoder parts (Model6,bleu4=0.3307).

Finally, we compared the pros and cons of all our models and discussed how these factors might affect our final performance.

1. Introduction

Multimodal machine learning, which combines information from multiple modalities such as text, image, and audio, has emerged as a promising research area in recent years. One of the most exciting applications of multimodal machine learning is image captioning. Image Captioning is the process of generating textual descriptions of an image. This task has great practical significance in areas such as assistive technology for the visually impaired, automated image indexing and retrieval, and personal digital assistants. The most common deep-learning based solutions address this task in two steps[8]: 1. a visual encoder model capable of generating the most relevant feature abstracts from a given image 2. a language model capable of generating meaningful and syntactically correct sentences. In this project, our team explored the combinations of various model structures and aim to build a good model of our own for the task of image caption.

The image caption task is a very classical task in computer vision and natural language processing. Some earliest works on image captioning were presented in 2015 by Vinyals et al.[9] at ICML2015 and by Andrej Karpathy et al[12] at CVPR2015, where both of them proposed a neural network-based model that consists of two main components: an encoder, which processes the input image and encodes it into a fixed-length vector representation, and a decoder, which generates the corresponding caption word by word based on the encoded representation of the image. In Vinyals's paper, the encoder is a CNN that is pre-trained on a large image classification dataset, while the decoder is a RNN that uses LSTM cells. Their results have similar performances with human, and has a BLEU-4=27.7 on COCO dataset.

Starting from 2015, the task has generally been addressed with pipelines composed of a visual encoder and a language model for text generation. The advancement includes: 1. Attention mechanisms allow the model to selectively attend to different parts of the input image when generating the caption. For example, the Show, Attend and Tell model [1] uses a visual attention mechanism to attend to different regions of the image while generating words in the caption. And the SCAN model by Lee et al. [7] uses a hierarchical attention mechanism to attend to both image regions and words in the caption. The SCAN model

achieved state-of-the-art performance on the COCO dataset at the time of publication. 2. Transformer-based models that can handle longer sequences of text than RNN-based models, and can better capture long-range dependencies in the image-caption pairs. For example, the ViLBERT model by Lu et al. [6] combines a BERT-based language model with a VisualBERT-based image encoder to generate captions for images. The model achieved state-of-the-art performance on the COCO dataset at the time of publication. 3.Vision-language pretraining models. The models are trained on large-scale datasets that combine both image and language data. Then, the pretraining models then can be used in the image caption tasks. For example, the ViLBERT model by Lu et al. [6] and the OSCAR model by Li et al. [11]. The OSCAR model pretrains a transformer-based language model on a dataset of over 4 million image-caption pairs. The pre-trained model can then be fine-tuned for image captioning. Their model has BLEU-4=41.7. 4. Multi-model learning: such as Kim et al. [3] uses cross-modal skip-connections between a vision transformer and a language transformer, and they achieve BLEU-4=46.5, which is currently state of the art based on https://paperswithcode.com/sota/image-captioning-on-coco-captions.

Despite all these advancements, detailed studies regarding the effect of the basic elements in these tasks, such as image feature extraction methods, effects of attention mechanisms, and architecture of transformers under similar task settings are rarely published.

In this project, we focus on the effect of different architectures and test their performances on COCO2014 datasets(train, val, test)[2, 10]. We modulize our architectures into visual-encoder and NLP-decoder, which allowed us the change the architecture of each module while keeping the other part unchanged. We built our the baseline architecture (Model1), with RESNET101 as the visual-encoder module, and LSTM as the NLP-decoder module. We also added attention mechanisms to attend.

We then tested the effect of a different visual-encoder architecture. In recent years, several non-CNN-based structures such as MLPMixer[4], an architecture based entirely on multi-layer perceptrons (MLPs) that are repeatedly applied across either spatial locations or feature channels. MLPMixer developped by Google show good performance in image classification tasks. However,not many researches have been done regarding using such model to do image caption tasks. We applied MLPMixer as an alternative visual-encoder, and built our Model2.

Then, we tested the effect of the NLP-decoder module architectures. We first test the effect of attention mechanisms by remove attention layers in Model1 to get our Model3.

In recent years, Transformer has become the state-of-the-art model to process natural language. Compared to traditional models such as RNN and LSTM, the Transformer model can process longer sequences more effectively and efficiently. Unlike RNN-based models that recursively rely on the last hidden state and the current input to generate the new output, transformer can process the input sequence all at once, which enables parallelism and process sequences more efficiently, while also sufferers much less from the vanishing gradient problem. Therefore, since our decoder is a language model, we decided to implement a Transformer model as well. We applied transformers in the NLP-decoder module. A transformer includes a sub-encoder and a sub-decoder part. We test the effects of different transformer structures by building models including 1. transformer with only the sub-encoder part (Model4); 2. transformer with only the sub-decoder part (Model5); 3. transformer with both sub-encoder and sub-decoder parts (Model6).

We trained all 6 models with similar settings, and evaluate their performances and compared the effect of those basic architecture elements based on COCO2014 caption dataset.Our work focuses on a very general model structure and a very common task in computer vision domain, hence we believe our work will be supportive to many researchers and scholars. Also, as we modulized our architectures and we built our different models in a stepwize manner, other researchers could easily reuse our models (or part of the models) in their work.

2. Approach

To implement our models, We studied code repositories on github, and found sgrvinod (https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Image-Captioning)[13] very instructive and helpful for our Model1, which is our baseline model. We updated the codes in the repository so we could run through the latest python package versions, and made necessary modifications so that the training curves of each epochs are logged.

We modulize our tasks into 5 modules:

1. Data loader module: We use Andrej Karpathy's training, validation, and test splits. During data reprocessing, we add `'<start>', '<end>'` tokens accordingly to each caption.

2. Model module: In this module, we implemented our proposed Model1-Model6 (details below)

3. Train module: In this module, we trained our proposed models. The matrix such as loss and accuracy were logged during training for model diagnosis. We use crossEntropy loss between predicted profiles and true captions. For the models with attention mechanisms including transformers, the attention weights(alpha) were also included in the loss for 'Doubly stochastic attention regularization' to prevent the attention mechanism from becom-

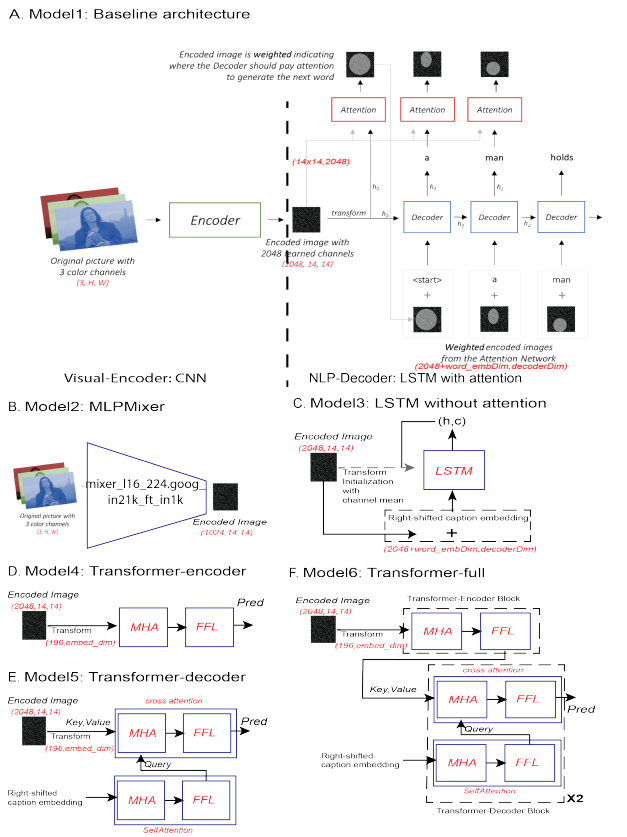

Figure 1. Architectures of Models we built.
(A). Model1, the architure have a visual-encoder module and a NLP-decoder moduel. The figure is adapted from source: https://github.com/RoyalSkye/Image-Caption. (B). Model2, the visual-encoder module in Model1 is replaced by MLPMixer Module. (C). Model3, the attention layers in Model1 are removed. (D). Model4, a transformer-encoder layer is used as the NLP-decoder. (E). Model5, a transformer-decoder layer is used as the NLP-decoder. (F). Model6, a complete transformer architecture is used as the NLP-decoder

ing too focused on a few input elements, or too spread out across many input elements, which can lead to poor performance. The bleu-4 scores during validation are also recorded. As a general guideline, we first trained our models without tuning the pretrained visual-encoder part for around 10 epochs or until there is no bleu-4 improvement in the validation dataset, and then we continue the training with visual-encoder tuning.

4. Evaluation module: In this module, for each image, we performed a Beam search for the best captions in the test set.

5. Visualization module: In this module, we generate a caption based on the Beam search and visualize the pixel attentions.

2.1. Model1

Model1 is our baseline model. It is a image caption pipeline with an encoder-decoder structure (Fig 1).

The visual-encoder is built on a pre-trained ResNet101 model. The last 2 layers of ResNet101 were removed and the forwarding features of the remaining layer were sent into a pooling layer before feeding into the decoder. The output of the visual-encoder is `(Batchsize,Width,Height,Channel)`.

The NLP-decoder is an LSTM-based attention layer. The hidden state (h,c) is initialized as a learn-able linear transformation of the channel mean of the visual-encoder. Both of which are `(Batchsize,channelMean,decoder_dim)` tensors.

Then the hidden state (h) and the visual-encoder output were then undergone the pixel attention layer to get the weights for each pixel, and the weight averaged channel mean for each visual-encoder output channel. This weight averaged image were combined with the TRUE caption as input for the LSTM cell to get renewed hidden state(h,c). Then, the hidden state (h) further go through full-connected layer to get the prediction scores for each vocabulary.

During fine-tuning training steps, we fine-tune the weights of encoder (except the weights of the first 5 layers) along with the decoder.

2.2. Model2:visual-encoder using MLPMixer

Based on Model1, We build the visual-encoder using a pre-trained MLPMixer and a decoder based on a LSTM-based Attention layer to get Model2. The output of the visual-encoder was the forward features of MLP-Mixer (which is the output after removing the original output layer designed for image classification jobs) and this matrix would then be sent into the downstream decoder and generate text.

During fine-tuning training steps, we fine-tune Fine-tune the last 4 Mixer Layers out of 16 in MLPMixer along with the decoder.

2.3. Model3:NLP-decoder using LSTM without attention

In order to evaluate the effectiveness of the attention mechanism over the image captioning task, we built a Model3 with LSTM decoder without attention.We removed the attention mechanism from Model1 by assigning equal weight for each pixel, and averaged the encoder output across all pixels for each channel, the weight averaged image in Model1 was replaced by this new averaged image as the initial hidden state for the LSTM-Decoder in Model3. Please note that since there is no attention, so no regularization of alpha weights were added to the cross entropy loss.

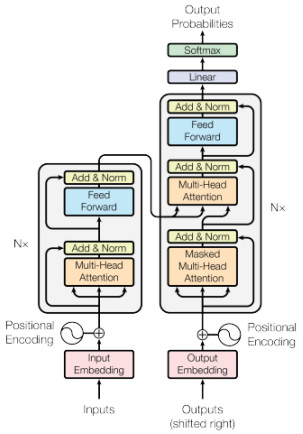

Figure 2. Transformer Architecture, adapted from 'Attention is all you need'(https://arxiv.org/abs/1706.03762)

Same as Model1, during fine-tuning training steps, we fine-tune the weights of encoder (except the weights of the first 5 layers) along with the decoder.

2.4. Model4:NLP-decoder using transformer-encoder layer

A complete transformer is composed of encoder blocks and decoder blocks (Shown in Figure 2). The original transformer is designed for seq-to-seq tasks such as translation task. In recent years, a lot of NLP models are based on this transformer architectures, such as Bert and GPT. BERT is a Transformer encoder whereas GPT is an autoregressive transformer decoder. Thus, it's worthwhile testing the effect of transformer encoder and transformer decoder in our image caption task.

In our Assignment4, we also implemented a transformer based seq2seq translation model, which outperformed both RNN and LSTM models. In that model, we use a Germany sentence as input, which goes through a self-attention multi-header layer and feed forward layer to predict the corresponding English translations.

Inspired by our Assignment4, we drafted our most simple transformer based decoder. In this structure, we treated our visual-encoder output as a 196-length, 2048-Embeded dimension sequence. This sequence then go to a TWO HEAD self-attention layer, and then feed forward layer to get the predictions. This prediction will finally go to the cross entropy loss with the true captions. Please note that, unlike other models, this model do not have teacher forcing, as the true captions only appears in the final loss function.

2.5. Model5:NLP-decoder using transformer-decoder layer

To implement the transformer decoder layer, we learned a lot from The RoyalSkye (https://github.com/RoyalSkye/Image-Caption)[5] repository, where a transformer-based image-captioning model is implemented. We carefully examined their code, integrated their code and updated them with our interfaces. Specifically, the right-shifted captions were embeded and go through a 2-Head self-attention layers. The outputs of this self-attention layers will go through an cross attention layers as a query vector, where the outputs of the visual-encoder layer act as a key-value vectors. In the final visualizations, the alpha weight were extracted from the final cross attention layer, and the attention figures were generated.

2.6. Model6:NLP-decoder using complete transformer

We further apply a complete transformer structure as the NLP-decoder. We tried to get better results by adding one more transformer-decoder blocks. In our final architecture, we use 2-header for all attention layers, 1 transformer-encoder blocks, and 2 transformer-decoder blocks.

2.7. Problems and solutions

The main problems that we anticipated are:

1. Adaptations between different modules in different model settings. As it is shown in Fig1, we modulize our models, which gave us much flexibility to replace the implementation methods for any module. However, it also brought the challenge that the returned values in different models might be different. To address this, despite checking all tensor dimensions, we also applied various tricks such resize, convolution layers,adptive_pool, etc. Specifically: 1. we add adptive_pool layer in Model1 to make sure we get a (14,14,2048) output in the visual-encoder module; 2. In MLPMixer, we resize the final output to match our visual-encoder module size; 3.In Model6, we need to forward visual-encoder outputs through a self attention layer. We applied position feed forward layer using Conv1d layers. 4. In Model5 and Model6, to match the dimentions of visual-encoder output with the embedding dimentions in the transformer, we applied a conv2D layer to convert the output channels of the visual-encoder output.

2. The transformer model requires a large memory and might be difficult to train with hardware limitations that we have. Luckily, all our training converged relative fast. On average, our computer could finish ONE epoch every hour (depending on whether visual-encoder tuning is turned on or not). Also, we decide to stop training if we observe not too much improvement in 3 epochs.

3. There are so many hyper-parameters to tune, and we have too many models to test. We need a balance. Some hyper-parameters such as learningRate, batchsize, loss regularizations could be tuned in different epochs. Some other hyper-parameters such as number of decoder layers, dimentions of word embeddings, number of attention heads, etc

could not be changed during the training epochs. Since our aim is to compare the effect of different architectures, we choose our optimal hyper-parameters based on their behavior in the first epoch. Also, we addapted a lr_scheduler ('torch.optim.lr_scheduler.ReduceLROnPlateau()'). Finally, we also applied a learning rate decaying strategy.

3. Experiments and Results

3.1. Measure Matrix

We used cross entropy loss as our training target. If there are are attention weights in the model, we also add a 'Doubly stochastic attention regularization' term to the loss.

During training, for each epoch, the training loss, training top5 accuracy, validation loss, validation top5 accuracy, and validation bleu-4 (Beamsize=1) values are also recorded. We monitor the training process based on the curves of these 5 parameters to tune hyper-parameters, to avoid overfitting, and to make sure we achieve a relative merged and stable status. We used the validation bleu-4 score to get the best trained model for each architecture.

Finally, for each model, we applied Beam Search (size=5) for each image in the test set, and get the bleu-4 score for each model based on the Search results.

3.2. Dataset

Images: We are using the MSCOCO'14 Dataset: Training(83K), Validation(41K), and Testing(41K).

3.3. Trainings

Figure 3 showed the training curves for each of our models based on top 5 accuracy. We chose this one to plot, mainly because we have slightly different definitions of loss function for each model because of attention weight regularization. Using top 5 accuracy would make the curves more comparable between different models. All of our models converged fast in a few epochs and then got stuck. Further tuning on the visual-encoder weight will mostly be helpful and the accuracy could further go up for a few epochs and then the training curve began to show signs of overfitting as indicated by no improvement in bleu-4 / loss_validation/acc_validations etc.

We noticed that overall speaking, with less parameters, the models will converge faster, as we need to tune the visual-encoder much earlier in Model3(Without attention) and Model4(2 Head Transformer-encoder as NLP-decoder).

3.4. Evaluation results

Table1 showed the bleu-4 score based on the test set.

3.5. Visualization

Except Model4, all other architectures could generate meaningful and relatively accurate image caption text as

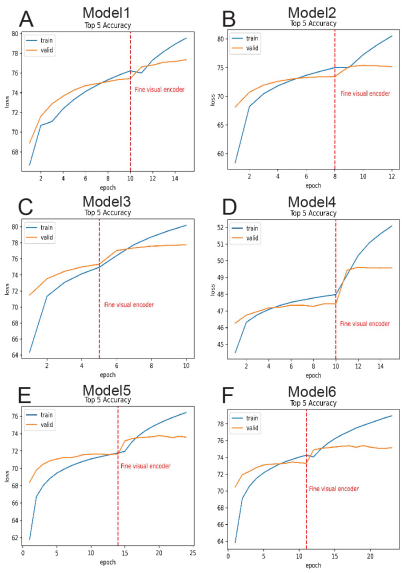

Figure 3. Training Curves

Table 1. bleu-4 scores based on Beam Search in the Test set (beamsize=5, on best model chosen by bleu-4 score in the validation set)

	Model1	Model2	Model3	Model4	Model5	Model6
bleu4	0.3372	0.2996	0.3388	0.1399	0.3210	0.3307

demonstrated in Figure4. Top captions generated using beam search (beam size=5) is listed on the right side of the demo figure. (B-G) showed how attention works. In (B,F), we could see that attention mechanism work very well to capture the related objects in the image. In (C,G), though the caption is relevant, but the attention weights are chaos. We examined and discussed this in the 'Discussions' section. In (D), because attention mechanism is removed, so there is no attention to the weights.

3.6. Discussions on our results

It's noticeable that Model4 is significantly worse than the other models. This seems a litter surprising to us at first, as similar architecture worked much better than RNN/LSTM based structures in Seq2Seq translation tasks in our Assignment4. However, after careful examine of our model architectures, the information flows, we noticed that Model4 is the only model that the true caption information is not used in the NLPdecoder module. In the LSTMbased NLPdecoders such as Model1,Model2,and Model3, the current 'True' caption word and information from the visual-encoder are combined to generate the predictions of the next word. In the Transformer-decoder based NLPdecoders such as Model5 and Model6, the caption is right-shifted, and tgt_mask during training to implement teacher forcing. The tgt_mask played a similar role as the repeated LSTM cells such that only word information before the prediction posi-

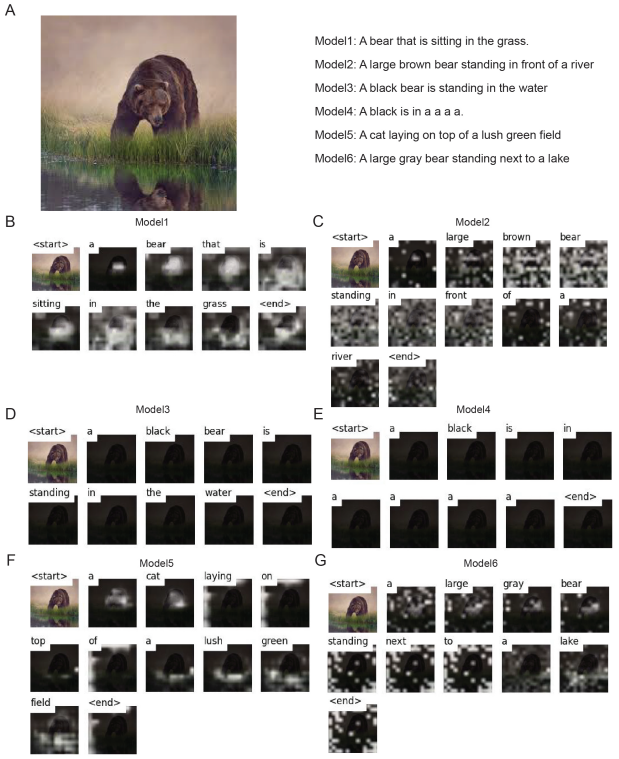

Figure 4. MLPMixer Sample Output

tion are used.

Only in our Model4, the true caption information is not used. We argue that we could get better results if we replaced the self-attention layer with an across-attention layer in Model 4. We tried this one, and indeed, this modified structure showed a bleu-4=0.3187, which is much higher than our current Model4 (bleu4=0.1399). This finding suggested that we need to design an architecture that could reflect reality. In our Assignment4, it's a Seq2Seq translation task, Germany and English are also very similar sequence languages, and almost every Germany word could be translated into some English words, thus seq(Germany)–¿ seq(English) architecture as in our Model4 is good enough. However, in our current task, the input is figure features from visual-encoders, which is not sequence information, a teacher forcing mechanism could largely improve the model training and model behavior.

Also, there is not too much difference between Model1 and Model3, suggesting that positional information in COCO2014 might not be that important given our evaluation matrix. This is because, in Model3, we only used the average value of each feature channel as the initial value for the hidden state of LSTM cells and the positional values were not used. We argue that if we really want to show the advantages of attention mechanism in this case, we might need image-caption set where captions are more related to the positional information. Still, you get what you trained on your training set.

Thirdly, in the transformer based NLP-decoders, we saw a slightly improvement in Model6, where a transformer-encoder block was added to further transform the visual-encoder outputs, comparing to Model5. One disadvantage of adding this layer is in the visualization step. As we are using cross attention weights for visualization. In Model6, these weights are the weights on the transformer-encoder output, our visualization looks a little weird comparing to Model5. We argue that this might because the pararmeters in the transformer-encoder are not optimized. If we could make a copy of weight in Model5, and only train the transformer-encoder part for several epochs before full tuning, we might be able to get much better visualizations.

Fourthly, the CNN based features seems to more relavant than MLPMixer output features as Model1 is better than Model2 in our task. Though the size of MLPMixer-based model is bigger and the inference speed is slower comparing the pipeline based on ResNet101. After reviewing the attention map generated by the decoder of the MLPMixer output (during the visualization process, we reshaped the output vector into a 14 by 14 matrix), we didn't find any clear pattern (Figure 4C), indicating that MLPMixer has very different conductive bias compared with CNN structure, the position-related information is either lost or encoded in a indirect and complex way.

Finally, the difference between LSTM-based NLP-decoders and transformer is also very small. Given the minor differences between Model1 and Model3, the positional information in the images are not that critical in COCO2014 dataset, thus it not surprising that transformer based models did not show too much advantages.

4. Further work

We submitted our codes with a Readme markdown file. Since our aim is the study the effect of different architecture elements in the image-caption task, there are a lot of other elements worth trying. such as:

1. Adding a pretrained word embeddings. Our current embeddings are learned from scratch. A well pretrained word embedding model might be helpful. Even better, if we could apply some Vision-language pretraining models.

2. Using ViT in the visual-encoder part, as we mentioned in the introduction part, some others get better results using ViT based models (For example, the ViLBERT model by Lu et al. [6] and the OSCAR model by Li et al.[11]).

3. Other tricks such as Multi-model learning, such as Kim et al. [3] uses cross-modal skip-connections between a vision encoder and NLP-decoder.

5. Work Division

Please add a section on the delegation of work among team members at the end of the report, in the form of a table and paragraph description. This and references do **NOT** count towards your page limit. An example has been provided in Table 2.

References

[1] Kelvin Xu at el. Show, attend and tell: Neural image caption generation with visual attention, 2016. 1

[2] Chen et al. Microsoft coco captions: Data collection and evaluation server, 2015. 2

[3] Chenliang Li et al. mplug: Effective and efficient vision-language learning by cross-modal skip-connections, 2022. 2, 6

[4] Ilya Tolstikhin et al. Mlp-mixer: An all-mlp architecture for vision, 2021. 2

[5] Jianan et al. Image-caption. https://github.com/RoyalSkye/Image-Caption, 2021. 4

[6] Jiasen Lu et al. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks, 2019. 2, 6

[7] Kuang-Huei Lee et al. Stacked cross attention for image-text matching, 2018. 1

[8] Matteo Stefanini et al. From show to tell: A survey on deep learning-based image captioning, 2020. 1

[9] Oriol Vinyals et al. Show and tell: A neural image caption generator, 2015. 1

[10] Tsung-Yi Lin et al. Microsoft coco: Common objects in context, 2014. 2

[11] Xiujun Li et al. Oscar: Object-semantics aligned pre-training for vision-language tasks, 2020. 2, 6

[12] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions, 2015. 1

[13] sgrvinod et al. Pytorch tutorial to image captioning. https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Machine-Translation, 2018. 2

Table 2. Contributions of team members
Note:
1. All members participate in all aspects of the project, including regular on-line meetings, coding/debug discussions, hyper-parameter testings, and writing reports;
2. Model1 is our basic architecture. All team members contribute to the understanding, coding and updating of Model1.
3. Most of the time, code implementations and training are separated, so that all team members could understand all aspects of our models.

Student Name	Contributed Aspects	Details
Kun Yu (kyu311)	Implement Model1 and Model3 Train and analyze Model4 Write the final report	1. Studied and shared the details of the attention mechanisms 2. Implemented models without attention mechanisms to study the effects of attention. 3. Studied and shared the "Doubly stochastic attention regularization" 4. Helped with the material collection and meeting archiving.
Quan Qi (qqi33)	Implemente Model1 and Model2 Train and analyze Model1 and Model2 Write the final report	1. Studied and shared with others the MPLMixer mechanisms. 2. Implemented MLPMixer visual-encoders. 3. Implement codes in the visualization module to visualize the attention area in the figure for Model2. 4. Update data loading module fit all Model requirements
Runfeng Zhang (rzhang668)	Implemente Model1, Model4 and Model6 Train and analysis Model3 Write the final report	1. Studied and shared the details of transformers. 2. Implemented transformer-encoder layer. 3. Merged Model4 and Model5 to get Model6. 4. Identified the limitations of Model4 which did not have a teacher forcing mechanism.
Yuan Shang (yshang42)	Implemente Model1, Model5 and Model6 Train and analysis Model 5 and Model6 Write the final report	1. Studied and shared the details of transformers. 2. Implemented transformer-decoder layer. 3. Merged Model4 and Model5 to get Model6. 4. Implement evaluation beam search for the transformer based NLP-decoders.