ContCap: A comprehensive framework for continual image captioning

Giang Nguyen\textsuperscript{1} Tae Joon Jun\textsuperscript{2} Trung Tran\textsuperscript{1} Daeyeong Kim\textsuperscript{1}
\textsuperscript{1} KAIST, Daejeon, South Korea \textsuperscript{2} Asan Medical Center, Seoul, South Korea
\textsuperscript{1}\{dexter.nguyen7, trungtq2019, kimd\}@kaist.ac.kr \textsuperscript{2}taejoon@amc.seoul.kr

Abstract

While cutting-edge image captioning systems are increasingly describing an image coherently and exactly, recent progresses in continual learning allow deep learning systems to avoid catastrophic forgetting. However, the domain where image captioning working with continual learning is not exploited yet. We define the task in which we consolidate continual learning and image captioning as continual image captioning. In this work, we propose ContCap, a framework continually generating captions over a series of new tasks coming, seamlessly integrating continual learning into image captioning accompanied by tackling catastrophic forgetting. After proving catastrophic forgetting in image captioning, we employ freezing, knowledge distillation, and pseudo-labeling techniques to overcome the forgetting dilemma with the baseline is a simple fine-tuning scheme. We split MS-COCO 2014 dataset to perform experiments on incremental tasks without revisiting dataset of previously provided tasks. The experiments are designed to increase the degree of catastrophic forgetting and appraise the capacity of approaches. Experimental results show remarkable improvements in the performance on the old tasks, while the figure for the new task remains almost the same compared to fine-tuning. For example, pseudo-labeling increases CIDEr from 0.287 to 0.576 on the old task and 0.686 down to 0.657 BLEU1 on the new task.

1. Introduction

Coming with the success of deep learning over the past decade, image captioning is rising as one of the most attractive domains because of its mouth-watering applications. Dealing with image captioning is to generate the most suitable caption describing the content of an image as much exact as possible. From the introduction of Neural Image Caption Generator (NIC) \cite{NIC}, various advanced techniques have been proposed to ameliorate image captioning \cite{VGG, VGG2, VGG3}. Experiments are frequently on three benchmark datasets: Flickr8 \cite{flickr8}, Flickr30 \cite{flickr30}, and MS-COCO \cite{coco}.

However, since image captioning models are built on the concept of Gradient-Based Neural Networks \cite{gradient}, they inevitably suffer from catastrophic forgetting. A trained model can work well in a specific task thanks to the distribution it learned before. Nonetheless, it struggles to make use of this model on a new task with remaining its performance on the old task. More precisely, the model could reach a high performance on the new task, but its performance will be degraded disastrously on the old task. Though novel models \cite{large, large2} greatly improve the performance of image captioning, they do not spend much concern on solving catastrophic forgetting.

Continual learning (CL) enables deep learning models to continuously learn from a series of dependent or even independent tasks with an acceptable degree of forgetting. Besides, fine-tuning is a popular way used in deep learning, where the model can utilize what it has learned before to start solving a new problem. When applying fine-tuning in image captioning, from Fig. 1 we can observe catastrophic forgetting by looking at the content of two generated sentences.

In Fig. 1 we first train a model on a set of 19 classes $S_{2-21}$ (class 2: bicycle to class 21: cow) - Task A. Notice that class 12 is missing in MS-COCO, so from class 2-21, we only have 19 classes. The model is an instance of convolutional neural network (CNN) + long-short term mem-
ory (LSTM) architecture, then after training, we obtain the model $M_{19}$. After that, we train on class 1: person $S_1$ - Task B, fine-tuning from model $M_{19}$ to get the model $M_{20}$. Now to observe catastrophic forgetting, we pick an image from class 18: dog in test set of Task A and perform inference with both $M_{19}$ and $M_{20}$. With $M_{19}$, we call original model, the description is: a black dog laying on a grass covered field. However, after fine-tuning to have $M_{20}$, the caption is shifted to: a little boy is standing in the grass with a frisbee. When learning the new task B, the model seriously forgets what it has learned before and seems to overemphasize describing person.

In this paper, we propose ContCap that is an extensive framework to combine encoder-decoder image captioning architecture and continual learning. To overcome catastrophic forgetting, we introduce a pseudo-label approach which is an extended version from LwF [13] as our method works on the multi-modal architecture, while LwF only works on CNN for image classification. The idea of LwF is to record the output of the previous model on the new data to train the new task. Two strategies of freezing (encoder or decoder) are also integrated to transfer the knowledge among tasks for preserving information of old tasks, while adapting to work well on the new ones. Furthermore, we employ knowledge distillation [9] on the output layer to guide the new model to produce the similar outcome to the old model.

Our paper addresses the most challenging scenario in continual learning while new training samples of different, previously unseen classes become available in subsequent tasks [21]. This scenario can be referred to as class-incremental learning [16]. In particular, no access to data from old tasks is permitted, ensuring privacy concerns or memory obstacles. To perform experiments on class-incremental setting, we create a new dataset from MS-COCO 2014 named Split MS-COCO in which each task coming with a new class. For example, with an old task A of classes cat, dog, table, the new tasks B will contain samples of class person only. When training task B, information from task A will be used to help the current model generalize well on all 4 classes cat, dog, table, person. We propose a terminology “clear image” meaning an image containing only one class in its annotation to manipulate incremental steps correctly. From over 82k images of training set and 40k of validation set from MS-COCO 2014, Split MS-COCO consists of over 51k and 23k respectively. Fine-tuning is considered as the baseline for comparison and evaluating effectiveness of strategies.

The experiments show catastrophic forgetting in image captioning and improvements of using proposed techniques. The previously learned tasks are well captioned, whilst new information is also well absorbed. Traditional metrics in image captioning BLEU [22], METEOR [2], ROUGE-L [14], and CIDEr [28] are calculated to quantitatively assessed each model.

As far as we know, this is the first work conducting image captioning in continual learning setup. Furthermore, we provide future directions and discussion over experiments to elaborate the results qualitatively and quantitatively.

**Contributions:** Our main contributions are:
- Our work is the first attempt considering addressing catastrophic forgetting in image captioning without the need for accessing data from existing tasks.
- We present ContCap - a comprehensive framework that reconciles image captioning and continual learning in the scenario of class-incremental. Our framework could be easily adapted to further research in continual image captioning or video captioning, and easily adapting with existing captioning models.
- Our experiments prove the manifestation of catastrophic forgetting in image captioning and our proposed methods help mitigate catastrophic forgetting in image captioning with no need of data from previous tasks. The experiments reveal the impact of frozen techniques, pseudo-label, and distillation in dealing with catastrophic forgetting in the context of image captioning.
- We propose a new dataset named Split MS-COCO from the standard MS-COCO dataset. This can be a reference for other new benchmark datasets for continual learning. The details of creating the dataset lie in the section 4.

2. Related work

**Image Captioning** The most simple and prevalent architecture of an image captioning system is based on encoder-decoder architecture. Basically, the features of an image are extracted as a result of a CNN suppressed its output layer. The intermediate features are then fed into an embedding block to synchronize with the input of a Recurrent Neural Networks (RNNs) in terms of data shape. The RNNs, often powered by an LSTM, is in charge of generating a sentence step by step conditioned on the current time step and the previous hidden state. Word generation is performed until we get the end token of the sentence. NIC [29] uses batch normalization and trains with maximum likelihood estimation. Xu et al. [31] propose a novel method imitating attention mechanism of human. The method allows us to focus on a certain region with high resolution while perceiving the surrounding in low resolution, and then adjust the focal point or do the inference accordingly. The idea of using attention mechanism is also applied in Neural Baby Talk [19] in a combination with template-based image captioning approach. The idea is to use an object detector to detect the visual words in an image, and then decide whether a textual word (word in vocabulary) or a visual word (detected
Continual Learning Continual learning requires the ability to learn over time without witnessing catastrophic forgetting or known as semantic drift, and allowing neural networks to incrementally solve new tasks. To achieve these goals of continual learning, studies mainly focus on addressing catastrophic forgetting problem. While the community has not agreed yet on a shared categorization for continual learning strategies, Maltoni et al. [20] proposes a three-way fuzzy categorization of the most common CL strategies: Architecture, regularization, and rehearsal. Rehearsal approaches [18][27][3] demand data from old tasks, leading to privacy or limited storage budget issues, making them non-scalable in the case of the number of classes is large. Architectural techniques [25][13], at the same time, alter the original network architecture for adapting to a new task besides preserving old memory. When a new task comes, layers or neurons are typically added to the previous model resulting in a new one, thus the capacity of the network are increased accordingly. This leads to a storage problem when the number of new classes is numerous because the final model could be gigantic, alleviating its portability. The concept of regularization approaches is to keep the model close to its previous state through adding a penalty in the objective function. Neither accessing old samples nor expanding the old network is required in regularization methods. Ideas from continual learning are widely adopted for vision tasks like object detection and image classification [26][24][3][30]. Very recently, Michieli et al. [21] introduce a novel framework to perform continual learning for semantic segmentation. Our work is mainly based on the idea of regularization but slight expansions in the decoder are needed to secure the preciseness of prediction.

Continual Learning in Vision Task So far, continual learning has been experimented on several vision benchmarks like Permuted MNIST [12], MNIST Split, and CIFAR10/100 Split [32] for image classification. Shmelkov et al. [26] use PASCAL VOC2007 [5] and COCO datasets [15] for object detection; and PASCAL VOC2012 [6] is selected in [21] for semantic segmentation.

3. Method

Given a task followed by a series of tasks, the ultimate target is to maximize the performance of captioning on the new task in addition to old images. The ContCap framework is presented in Fig. 2. At any time point, the framework has performed a sequence of learning tasks, and at time t, samples from dataset of task $T_t$ are previously unseen with the other tasks. In learning task $T_t$, continual learning techniques are applied with information from Knowledge Base (KB) with an expectation to describe well on both the task $T_t$ and the previous ones. KB contains the acquired knowledge and the knowledge accumulated from learning the old tasks.

The proposed approaches could be applied on any encoder-decoder architecture-based image captioning models, but to evaluate, we pick the most simple architecture consisting of CNN + LSTM. The encoder for feature extraction is the pre-trained ResNet-152 [8]. For decoder part, we build a single layer LSTM network for managing the process of description generation. A task $T$ is denoted by its classes $C$, data $S$, obtained model $M$ and vocabulary $V$. Initially, we train the network in a set of classes $C_0$ with the corresponding training data $S_0$ to get the model $M_0$. Next, with incremental steps $k = 1, 2, 3, ..., $ in task $T_k$, we load the model $M_{k-1}$ and update the current model employing the distribution of $S_k$ like in [21].

Although tasks coming later are independent from previous tasks, a vocabulary should be accumulated and transferred progressively. For example, in an early task of 20 classes, we have a vocab $V_{20}$ of $v_{20}$ entries. While a task of person comes, we simply perform tokenization by NLTK [17] on the annotations of samples, then pick the most significant words into the vocabulary. The accumulated vocabulary so far is $V_{21} = V_{20} \cup V_{\text{person}}$, and the number of entries in the vocabulary is $v_{21} = v_{20} + v_{\text{person}} - v_{V_{20}} \cap V_{\text{person}}$.

We run the fine-tuning experiment at the first step to get the baseline for comparison. With new classes, we initialize model parameters by the converged state of an old task without adding any techniques. Predictably, the new model will try to generalize on the newly added classes and give a poor performance on the old classes.

3.1. Freezing

A heuristic solution to hinder catastrophic forgetting is to freeze a part of model. As our architecture is divided into
encoder and decoder, keeping one component intact could affect directly to the performance of the previous tasks as well as the new task.

![Figure 3. Freezing a component of fine-tuned model while the remaining component is trainable.](image)

On the one hand, encoder is frozen and decoder is updated exploiting the new dataset. The feature extraction stays unchanged compared to the previous model \(M_A\) which plays a role of fine-tuned model (Task A), and this model is absolutely frozen. The feature extractor \(E\) is constrained, leaving dealing with new task for the decoder \(D\), we refer as \(E_{F}\) (see Fig. 3). Different from \([21]\) while new nodes are added to the decoder for segmentation task based on the number of new classes present, the decoder here are modified according to the vocabulary size. The vocabulary is expanded over tasks to ensure the naturalness of the pre-

On the other hand, we freeze decoder \(D\) while encoder is trainable. Because neurons are added to decoder as by virtue of expanding vocabulary, the newly attached neurons are the only trainable part of decoder. By setting this, the convolutional network can better generalize features, thus feeding more fine-grained input for the decoder. In the case of both \(E\) and \(D\) are frozen, the new model is the same as the previous model, while fine-tuning is the case that both \(E\) and \(D\) are trainable.

3.2. Pseudo-labeling

We refer this method as pseudo-labeling since when training, pseudo-labels are generated to guide the current model to mimic the behavior of the previously trained model. The procedure of this approach is described as Fig. 4.

![Figure 4. Procedure of pseudo-labeling.](image)

Pseudo labels \(Y_{k-1}\) are acquired by inferring the caption of all input images in the dataset of the new task using the previous model. Differ from \([13]\), the convolutional network architecture stays unchanged during training although new classes appear since the expected output is textual caption, not the probabilities for each class as in object detection or semantic segmentation. The model is then trained in a supervised way to minimize the loss computed based on the ground truth \(Y_k\), the predicted caption \(\hat{Y}_k\), and the pseudo labels. The number of captions per an image - cri is 5 following the standard MS-COCO dataset, and \(L\) is the cross-entropy loss function. The loss component from pseudo-labeling \(L_P\) is explicitly written as:

\[
L_P = -\beta \cdot Y_{k-1} \log \hat{Y}_k
\]

where \(\beta\) plays a role as a regulator to accentuate the old or new task.

3.3. Knowledge Distillation

Knowledge distillation works on a teacher-student strategy when we want to transfer the knowledge from a teacher model to a student model. Training the student model tries to minimize the difference between the predictions of the teacher and the student. By applying distillation, teacher is the model obtained from the old task, when student is the new model. One image \(X_n\) is passed through both teacher and student, then the mean squared error between outputs \((Y_{st} - Y_{tr})\) is added to the loss function of the student to penalize it. The distillation term \(L_{dis}\) is computed as:

\[
L_{dis} = \lambda \cdot \|Y_{tr} - Y_{st}\|^2
\]

where \(\lambda\) could be increased to encourage student more intensively to imitate the behaviors of teacher.

3.4. Objective

The objective function can be rewritten by \(L_{CE}, L_P\) and \(L_{dis}\) in which \(L_{CE}\) is the cross-entropy loss over annotations and the prediction, \(L_P\) is the additional loss from pseudo labeling, and \(L_{dis}\) is the penalty from teacher on student model.

\[
\mathcal{L} = L_{CE} + L_P + L_{dis}
\]

\[
= -\sum_{i=1}^{5} Y_i^k \log \hat{Y}_k - \beta \cdot Y_{k-1} \log \hat{Y}_k + \lambda \cdot \|Y_{tr} - Y_{st}\|^2
\]
To control the impact of pseudolabels or distillation in training, $\beta$ or $\lambda$ is adjustable, and in case of $\lambda = 0$, meaning we are training the new model on a fine-tuning scheme. In our experiments, $\beta$ and $\lambda$ are mostly set to 1 and 10 respectively. Increasing $\beta$ or $\lambda$ will favor the old task over the new one.

3.5. Implementation Details

Feature extraction We use pre-trained model of ResNet-152 \cite{8}, removing the final fully connected layer to obtain the immediate features. Behind the ResNet is an embedding layer which has an embedding size of 256. This embedding layer is responsible for bridging the output of the ResNet and the input of the RNN, followed by a batch normalization layer with $momentum = 0.01$.

Language model The entrance of the language model is an embedding module to embed the features and labels to feed the LSTM. The RNN contains a single layer LSTM with the size of hidden state is 512 and the embedding size is 256. A fully connected layer is placed at the end of the language model to generate the score for each entry in the vocabulary. Note that when the vocabulary is increasingly accumulated, the embedding layer and the output layer of the decoder are also enlarged accordingly.

Training details To help the model work with images of different sizes, we always resize input to square images (224 × 224). We shuffle images at each epoch, and the batch size is 128. Early stopping is used to help models reach their optimal state within 50 epochs of training. The value patience - the number of epochs to wait before early stop if no loss improvement on the validation set is empirically set to 5. For updating both encoder and decoder, we choose Adam optimizer \cite{11} with a shared learning rate of $5 \times 10^{-4}$. Exceptionally, in knowledge distillation, we apply another Adam optimizer on encoder with a learning rate of $5 \times 10^{-5}$. The parameters of encoder could change dramatically during training time, so we want to slow it down.

All methods but distillation are initialized by the final state of the old task. While joint training is incorporated only in pseudo-labeling, normal training procedure is coupled to each remaining technique. Joint training is when we first freeze the entire model except for the newly added neurons, then train to converge. Subsequently, we unfreeze the model and perform training again until convergence (joint-optimization).

4. Experimental Results

Our experiment results have been recorded on Split MS-COCO, a variant of the MS-COCO benchmark dataset \cite{15}. The original MS-COCO dataset contains 82,783 images in the training split and 40,504 images in the validation split. Each image of MS COCO dataset belongs to at least one of 80 classes. To make Split MS-COCO, we perform processing on MS-COCO as the following steps:

1. Split all images into distinguished classes. There are 80 classes in total.
2. Resize images to a fixed size of (224 × 224) in order to deal with different sizes of input.
3. Discard images including more than one classes in its annotations to obtain “clear” images only.

Since MS-COCO test set is not available, we divide the validation set of each class into two equal parts: one is for validation and the other for testing. If the number of images of a class is odd, the number of images in the validation set will be one image larger than that of the test set. Finally, we get 47,547 images in the training set, 11,722 images in the validation set, and 11,687 images in the test set.

We assess the learning capacity of the framework by commonly used metrics such as BLEU, ROUGE, CIDEr, and METEOR. $F$, $E_F$, $D_F$, $P$, and $KD$ are fine-tuning, freezing encoder, freezing decoder, pseudo-labeling, and knowledge distillation respectively. Fine-tuning is incorporated in almost the remaining techniques by default. These metrics generally measure the word overlap between the predicted and ground truth captions. Captions are inferred on every settings to qualitatively evaluate the framework (See Fig. 5).

4.1. Addition of One Class

In the first setting, we examine catastrophic forgetting by an addition of one class. We train the model $M_{19}$ with the training set of 19 classes (2 to 21) - $S_{19}$, and then evaluate on the corresponding test set of these 19 classes. Next, we analyze the learning capacity of the model by adding another class person and see how confident the model $M_{20}$ can operate on the old and new classes. The results are shown in Table 1.

From the table, we can argue that fine-tuning seriously demolishes the accuracy on the old task although we just add only one class. Massive decreases can be observed in all the metrics with a remarkable one observed in CIDEr, a drop of 0.778 to 0.287. On the old task, pseudo-labeling outperforms the remaining ones, reaching 0.576 CIDEr, which doubles as fine-tuning’s CIDER score. Freezing encoder and decoder achieve 0.409 and 0.461 CIDEr respectively. We conclude that freezing does not help much because when a part is frozen, the other will be strongly driven to the new domain, thus misdescribing images of the old domain. Pseudo-labeling improves retaining the old knowledge as the pseudo-labels impede the drift of model weights by the regularization. Knowledge distillation performs worst because we rebuild the model, not any infor-
mation is preserved from beginning, resulting in only 0.255 CIDEr.

When testing on the new domain, fine-tuning reaches a relatively high performance (0.686 and 0.713 for BLEU1 and CIDEr respectively). Simply freezing encoder, surprisingly, almost equalizes the performance of fine-tuning on the old task, reaching 0.685 BLEU1 and 0.702 CIDEr, while freezing decoder’s CIDEr is merely 0.467. Pseudo-labeling yields 0.654 BLEU1 and 0.467 CIDEr, and distillation claims 0.706 BLEU1 and 0.513 CIDEr.

Despite of an average performance on the new task of pseudo-labeling, this technique is a reasonable option when adding a new class as it balances the performance on the new task and old tasks the most. Freezing encoder is not balanced over all tasks, but it keeps the original performance on the new one while still diminished catastrophic forgetting.

4.2. Addition of Multiple Classes at Once

We assume the dataset from 5 classes person, sports ball, tv, toilet, and bottle is $S_{\text{multiple}}$. We choose those 5 classes because they are the biggest classes that are not existing in $S_{19}$, and they are from different super-classes [15], ensuring totally new instances are introduced. In this setting, starting with $M_{19}$ with $S_{19}$, we add $S_{\text{multiple}}$ at once to have $S_{24}$. After training to get the model $M_{24}$, we test the model on $S_{19}$ and $S_{\text{multiple}}$.

We first test $M_{24}$ over $S_{19}$. As the vocabulary size is broadened significantly due to new words coming from the annotations of the whole 5 new classes, the generated sentences take a chance to be more natural and diverse. As a result, all approaches in this experiment performs better than the ones when only one class is added. On $S_{\text{multiple}}$, the trend is similar to the first experiment when we add a new class person. We suppose that since the 5 new classes belong to different and non-existing super classes, the features from each of them are unique. We call this type of adding as “unknown-domain addition”. Adding multiple classes simultaneously is similar to adding a totally new class but generates more fine-grained descriptions thanks to a bigger vocabulary. While pseudo-labeling still has the performance balance (0.612 on $S_{19}$ and 0.572 CIDEr on $S_{\text{multiple}}$), distillation seems to perform the best in this scenario, at 0.436 and 0.785 CIDEr.

| $S_{19}$ | $S_{\text{multiple}}$ |
|----------|----------------------|
| M$_{19}$ | BLEU1: 0.681, BLEU4: 0.249, METEOR: 0.234, ROUGE$_L$: 0.508, CIDEr: 0.778 |
| F      | BLEU1: 0.519, BLEU4: 0.109, METEOR: 0.150, ROUGE$_L$: 0.386, CIDEr: 0.287 |
| $E_F$  | BLEU1: 0.563, BLEU4: 0.138, METEOR: 0.172, ROUGE$_L$: 0.414, CIDEr: 0.409 |
| $D_F$  | BLEU1: 0.587, BLEU4: 0.146, METEOR: 0.177, ROUGE$_L$: 0.420, CIDEr: 0.461 |
| P      | BLEU1: **0.617**, BLEU4: **0.188**, METEOR: **0.203**, ROUGE$_L$: **0.462**, CIDEr: **0.576** |
| $KD$   | BLEU1: 0.554, BLEU4: 0.099, METEOR: 0.143, ROUGE$_L$: 0.387, CIDEr: 0.255 |

Table 1. Performance of $M_{20}$ on $S_{19}$ and $M_{20}$ on the new class person.

| $S_{19}$ | $S_{\text{multiple}}$ |
|----------|----------------------|
| M$_{19}$ | BLEU1: 0.681, BLEU4: 0.249, METEOR: 0.234, ROUGE$_L$: 0.508, CIDEr: 0.778 |
| F      | BLEU1: 0.560, BLEU4: 0.141, METEOR: 0.167, ROUGE$_L$: 0.415, CIDEr: 0.361 |
| $E_F$  | BLEU1: 0.582, BLEU4: 0.159, METEOR: 0.184, ROUGE$_L$: 0.433, CIDEr: 0.445 |
| $D_F$  | BLEU1: 0.605, BLEU4: 0.179, METEOR: 0.191, ROUGE$_L$: 0.448, CIDEr: 0.529 |
| P      | BLEU1: **0.630**, BLEU4: **0.201**, METEOR: **0.207**, ROUGE$_L$: **0.470**, CIDEr: **0.612** |
| $KD$   | BLEU1: 0.594, BLEU4: 0.157, METEOR: 0.180, ROUGE$_L$: 0.434, CIDEr: 0.436 |

Table 2. Performance of $M_{24}$ on $S_{19}$ and $M_{24}$ on $S_{\text{multiple}}$ (classes person, sports ball, tv, toilet, and bottle are added at once).

| $S_{19}$ | $S_{\text{multiple}}$ |
|----------|----------------------|
| M$_{19}$ | BLEU1: 0.681, BLEU4: 0.249, METEOR: 0.234, ROUGE$_L$: 0.508, CIDEr: 0.778 |
| F      | BLEU1: 0.475, BLEU4: 0.06, METEOR: 0.114, ROUGE$_L$: 0.343, CIDEr: 0.099 |
| $E_F$  | BLEU1: 0.522, BLEU4: 0.107, METEOR: 0.152, ROUGE$_L$: 0.388, CIDEr: 0.264 |
| $D_F$  | BLEU1: **0.545**, BLEU4: **0.138**, METEOR: **0.160**, ROUGE$_L$: **0.407**, CIDEr: **0.359** |
| P      | BLEU1: 0.460, BLEU4: 0.073, METEOR: 0.122, ROUGE$_L$: 0.354, CIDEr: 0.152 |
| $KD$   | BLEU1: 0.498, BLEU4: 0.059, METEOR: 0.112, ROUGE$_L$: 0.361, CIDEr: 0.047 |

Table 3. Performance of $M_{24}$ on $S_{19}$ and $M_{24}$ on $S_{\text{multiple}}$ (classes person, sports ball, tv, toilet, and bottle are added at one by one).
Figure 5. Qualitative results on samples from the old and new task. Three scenarios are presented here showing the qualitative results are correlated with the computed metrics. The impacts of adding one class and multiple classes at once are approximately the same, while the quality of caption in sequential addition declines catastrophically after each step.

4.3. Addition of Multiple Classes Sequentially

Again, we begin with $M_{19}$ and $S_{19}$. After that, we add 5 classes: person, sports ball, tv, toilet, and bottle one by one. At each stage, the previous model is used to initialize the current model. The class of bottle is trained at the end of the process. Finally, we have $M_{seq}$ to manipulate evaluation on $S_{19}$ and $S_{multiple}$.

We realize that this setting causes extremely poor performance on $S_{19}$ regardless of the presence of the techniques. BLEU1 score which is computed in 1-gram manner is about 0.5, but other metrics collapse because they count matching n-grams or even more complex patterns. Sequentially adding new classes drifts the original model to various directions after each step, thus changing it drastically (stage catastrophic forgetting). CIDEr changes from 0.778 to only about 0.05 with distillation. Regarding to $S_{multiple}$, the performance is much lower than the figure for the setting of adding 5 classes once. Simultaneously adding helps model better generalize the knowledge on new classes, while sequentially adding suffers from both stage catastrophic forgetting and poor generalization. Freezing seems to govern the sentence generation process the best in this scenario.

5. Discussion

In this paper, we introduce ContCap, a comprehensive framework fusing continual learning and image captioning, working on a new dataset Split MS-COCO created from MS-COCO but working in continual learning settings. We firstly perform image captioning in incremental schemes and then add techniques from continual learning to weaken catastrophic forgetting. The experiments on three settings show the sign of catastrophic forgetting and the effectiveness when integrating freezing, pseudo-labeling, and distillation compared to fine-tuning. Applying further advanced techniques in continual learning to enhance the performance especially in the sequential addition setting is left for future research. Since image captioning task is multi-modal, further approaches should be adopted to reach a high performance on the new task like single-modal tasks (object segmentation or detection). As we refer “unknown-domain addition”, experiments in “known-domain addition” should be performed in an expectation of witnessing less forgetting compared to “unknown” scenario. Stage catastrophic forgetting should be more deeply investigated to make continual learning possible when we have a stream of new tasks.
References

[1] J. Aneja, A. Deshpande, and A. G. Schwing. Convolutional image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5561–5570, 2018.

[2] S. Banerjee and A. Lavie. 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, pages 65–72, 2005.

[3] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In Proceedings of the European Conference on Computer Vision (ECCV), pages 233–248, 2018.

[4] M. Cornia, L. Baraldi, and R. Cucchiara. Show, control and tell: A framework for generating controllable and grounded captions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8307–8316, 2019.

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338, 2010.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[7] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211, 2013.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[9] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[10] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[11] D. P. Kingma, J. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabyska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017.

[12] Z. Li and D. Hoiem. Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence, 40(12):2935–2947, 2017.

[13] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pages 74–81, 2004.

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014.

[15] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural
image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[32] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.