MixGen: A New Multi-Modal Data Augmentation

Xiaoshuai Hao*
Institute of Information Engineering, CAS

Yi Zhu
Amazon Web Services

Srikar Appalaraju
Amazon Web Services

Aston Zhang
Amazon Web Services

Wanqian Zhang, Bo Li
Institute of Information Engineering, CAS

Mu Li
Amazon Web Services

Abstract

Data augmentation is a necessity to enhance data efficiency in deep learning. For vision-language pre-training, data is only augmented either for images or for text in previous works. In this paper, we present MixGen: a joint data augmentation for vision-language representation learning to further improve data efficiency. It generates new image-text pairs with semantic relationships preserved by interpolating images and concatenating text. It’s simple, and can be plug-and-played into existing pipelines. We evaluate MixGen on four architectures, including CLIP, ViLT, ALBEF and TCL, across five downstream vision-language tasks to show its versatility and effectiveness. For example, adding MixGen in ALBEF pre-training leads to absolute performance improvements on downstream tasks: image-text retrieval (+6.2% on COCO fine-tuned and +5.3% on Flicker30K zero-shot), visual grounding (+0.9% on RefCOCO+), visual reasoning (+0.9% on NLVR2), visual question answering (+0.3% on VQA2.0), and visual entailment (+0.4% on SNLI-VE).

1. Introduction

Recent years have witnessed an explosion in vision-language research [31, 42, 3, 21, 37, 14, 48, 1]. In joint modality learning, models extract rich information across modalities to learn better latent representations. These models, however, are often trained on a massive number of image-text pairs using thousands of GPUs. For example, CLIP [38] matches the accuracy of ResNet-50 [13] on ImageNet by only using zero-shot, but it is trained with 400M image-text pairs for 12 days on 256 V100 GPUs. Furthermore, most of these large-scale datasets [38, 15, 63] are not publicly accessible. Even if they are available, reproduction and further improvement on existing methods are challenging for researchers with limited computing resources.

Data augmentation is widely used in deep learning to improve data efficiency and provide explicit regularization during model training in both computer vision (CV) [9, 65, 6, 64, 7, 68, 46] and natural language processing (NLP) [66, 49, 16, 54, 51, 5, 40, 59]. Applying existing data augmentation techniques to vision-language learning, however, is not straightforward. In an image-text pair, both the image and the text contain rich information that matches each other. Intuitively, we hope that their semantics still match after data augmentation. For example, consider an image with its paired sentence “a white dog playing in the right corner of the green lawn”. Applying data augmentation methods such as cropping, color changing, and flipping to this image may require changing the color and positional words in its paired sentence at the same time.

In order to preserve the semantic relationship, previous works perform mild data augmentation on either vision or text modality. ViLT [19] and following works [48, 23] adopt RandAugment [7] for image augmentations without color inversion. CLIP [38] and ALIGN [15] only use random resized crop without other image augmentations. In terms of language, most literature just leaves text data augmentation to be handled by masked language modeling [8]. There are works using co-augmentation [45, 58], but only designed for specific downstream tasks, rather than generic vision-language pre-training.

In this work, we introduce a multi-modal joint data augmentation method for pre-training: Mix Generation (MixGen). As shown in Figure 1, MixGen generates a new training sample by linearly interpolating images and concatenating text sequences from two existing image-text pairs. We can see that most objects and scene layout remain in the blended image, while the text information is fully preserved. The semantic relationship within the newly generated image-text pair is expected to match in most cases.

*Work done during an internship at AWS. First three authors contribute equally.
Thus, we can use the augmented data to improve vision-language model training. Despite its simplicity, using MixGen on top of strong baselines (e.g., ALBEF [23]) consistently improves state-of-the-art performance across five downstream vision-language tasks: image-text retrieval (+6.2% on COCO fine-tuned and +5.3% on Flicker30K zero-shot), visual grounding (+0.9% on RefCOCO+), visual reasoning (+0.9% on NLVR2), visual question answering (+0.3% on VQA2.0), and visual entailment (+0.4% on SNLI-VE). MixGen also leads to enhanced data efficiency, e.g., the performance of ALBEF with MixGen when pre-trained on 1M/2M/3M samples match baseline ALBEF pre-trained on 2M/3M/4M samples, respectively. In addition, we perform extensive ablation studies to understand the effects of various design choices in MixGen. Finally, MixGen can be incorporated into most methods with only a few lines of code. In terms of fine-tuned image-text retrieval on COCO, MixGen brings absolute improvements on four popular and varied architectures: ViLT (+17.2%), CLIP (+4.1%), ALBEF (+7.0%) and TCL (+3.2%).

2. MixGen

In this section, we introduce our multi-modal joint data augmentation technique: Mix Generation (MixGen). Suppose that we have a dataset of $N$ image-text pairs, where images and text are denoted as $I$ and $T$ with subscripts, respectively. Given two image-text pairs $(I_i, T_i)$ and $(I_j, T_j)$ for any $i, j \in \{1, \ldots, N\}$ and $i \neq j$, a new training sample $(I_k, T_k)$ is generated via

\[
I_k = \lambda \cdot I_i + (1 - \lambda) \cdot I_j \\
T_k = \text{concat}(T_i, T_j),
\]

where $\lambda$ is a hyper-parameter between 0 and 1, indicating linear interpolation between raw pixels of two images $I_i$ and $I_j$; and the concat operator directly concatenates two text sequences $T_i$ and $T_j$ to best preserve original information. In this way, the semantic relationship within the newly generated image-text pair $(I_k, T_k)$ still holds in most scenarios, such as in Figure 1 and Figure 2.

![Figure 1. Our proposed multi-modal data augmentation technique: MixGen. Given any two image-text pairs $(I_i, T_i)$ and $(I_j, T_j)$, we interpolate the two images and concatenate two text sequences, to generate a new image-text pair $(I_k, T_k)$. Figure best viewed in color.](image)

The pseudocode of MixGen data augmentation is presented in Algorithm 1. Given a minibatch of $B$ randomly sampled image-text pairs, MixGen replaces the first $M$ training samples with the newly generated pairs. Hence, the batch size, total training iterations and overall training pipeline remain the same. By default, we set $\lambda = 0.5$ and $M = B/4$ in Algorithm 1. Such a plug-and-play technique can be easily incorporated into most vision-language representation learning approaches and tasks: all it takes is just a few lines of code with minimal computational overhead. Take ALBEF [23] as an example, adding MixGen on top of it during pre-training only increases its overall wall clock training time by 0.4%.

2.1. MixGen variants

MixGen is in a very simple form (Algorithm 1). However, there could be multiple variants depending on how image and text augmentations are performed. Theoretically, we could also use other augmentations for images beyond mixup, and other text augmentations beyond concatenation, but the design space will be intractable. Thus, we focus on using mixup for image and concatenation for text, and select 5 most straightforward variants of MixGen to support our final design choice.

Since default MixGen takes a fixed $\lambda$, we introduce variant (a) with $\lambda \sim \text{Beta}(0.1, 0.1)$, following original mixup [65] that samples $\lambda$ from a Beta distribution. In order to show the benefit of performing joint image-text augmentation, we propose variant (b) and (c). To be specific,
variant (b) mixup two images and uniformly pick one text sequence, while variant (c) concatenates two text sequences and uniformly pick one image. In the end, we study whether we should use a subset of tokens, instead of concatenating all tokens from two text sequences. Variant (d) takes tokens from two text sequences proportionally based on a similar as image mixup and then concatenate. The other variant (e) first concatenates all the tokens, but randomly keep half of them to generate a new text sequence.

More detailed definitions of these 5 variants can be seen in Table 6. We also perform extensive ablation studies on them. As we will see in Table 7, our default MixGen achieves the overall best performance, and consistently outperforms other variants across four different vision-language downstream tasks.

### 2.2. Input-level and embedding-level MixGen

Another design perspective is where to apply the data augmentation. The formulation in Equation (1) is directly performed on the raw input, e.g., images and text sequences. Alternatively, the idea of MixGen can be applied on the embedding level. To be specific, instead of interpolating raw image pixels, we can interpolate image features that are extracted from an image encoder. Similarly, instead of concatenating two text sequences, we can concatenate two sequence features that are extracted from a text encoder. Denoting the training pairs with respect to their embedding as \((f_I, f_T)\) and \((f_{I'} , f_{T'})\), the newly generated training pair in its embedding form is

\[
\begin{align*}
\tilde{f}_I &= \lambda \cdot f_I + (1 - \lambda) \cdot f_{I'} \\
\tilde{f}_T &= \text{concat}(f_T, f_{T'}).
\end{align*}
\]

We call MixGen that is performed on the raw input as \textit{input-level MixGen}, and that performed on the embedding level as \textit{embedding-level MixGen}. As we will show in Figure 6 left, input-level MixGen consistently performs better than embedding-level MixGen. In addition, input-level MixGen has an advantage of implementational simplicity, since we don’t need to modify network architecture nor touch model forward. Hence, we refer to \textit{input-level MixGen} as our proposed MixGen for the rest of the paper unless otherwise stated.

### 3. Experiments

In this section, we first describe model pre-training in Sec. 3.1. Then we introduce five downstream tasks (image-text retrieval, visual question answering, visual entailment, visual reasoning and visual grounding) and present experimental results of using MixGen in Sec. 3.2 and Sec. 3.3. In the end, we show visualizations in Sec. 3.4 to demonstrate the benefits of using MixGen.

#### 3.1. Pre-training

**Pre-training methods** Data augmentation techniques are usually applicable to various models, because they are agnostic to network architectures, training objectives, optimizers, or learning schedules. In this study, we plug-and-play MixGen on four recent popular approaches with varied architectures, CLIP [38] (dual-encoder), ViLT [19] (single fusion encoder), ALBEF [23] (dual-encoder followed by fusion encoder) and TCL [60] (dual-encoder followed by fusion encoder). Also note, the pre-training objectives for these approaches are varied. We emphasize that MixGen can be easily incorporated to other vision-language pre-training methods with only a few lines of code. To be specific, once we get a minibatch out of the dataloader, we apply MixGen to generate new image-text pairs and update the minibatch. We then forward the updated batch to the network without modifying any of their original training settings.

**Pre-training datasets** There are four widely adopted datasets for vision-language model pre-training, includ-
Table 1. Fine-tuned image-text retrieval on Flickr30K and MSCOCO datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Images</th>
<th>Flickr30K (1K test set)</th>
<th>MSCOCO (5K test set)</th>
<th>Flickr30K (1K test set)</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Text Retrieval</td>
<td>Image Retrieval</td>
<td>Text Retrieval</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>R@1 R@5 R@10 R@1 R@5 R@10</td>
<td>R@1 R@5 R@10 R@1 R@5 R@10</td>
<td>R@1 R@5 R@10 R@1 R@5 R@10</td>
<td></td>
</tr>
<tr>
<td>UNITER-base [3]</td>
<td>4M</td>
<td>64.4 87.4 93.1 50.3 78.5 87.2</td>
<td>460.9</td>
<td>85.9 97.1 98.8 72.5 92.4 96.1</td>
<td>542.8</td>
</tr>
<tr>
<td>VILLA-base [10]</td>
<td>4M</td>
<td>70.0 91.1 95.5 54.0 80.8 88.5</td>
<td>479.9</td>
<td>86.6 97.9 99.2 74.7 92.9 95.8</td>
<td>547.1</td>
</tr>
<tr>
<td>OSCAR-base [27]</td>
<td>4M</td>
<td>70.5 91.5 97.2 54.7 82.2 89.2</td>
<td>488.0</td>
<td>89.7 98.4 99.1 74.7 93.4 96.1</td>
<td>551.4</td>
</tr>
<tr>
<td>UNIMO-base [26]</td>
<td>4M</td>
<td>- - - - - -</td>
<td>- -</td>
<td>- - - - - -</td>
<td>-</td>
</tr>
<tr>
<td>ViLT-base [19]</td>
<td>4M</td>
<td>61.5 86.3 92.7 42.7 72.9 83.1</td>
<td>439.2</td>
<td>83.5 96.7 98.6 64.4 88.7 93.8</td>
<td>525.7</td>
</tr>
<tr>
<td>ALBEF-base [23]</td>
<td>4M</td>
<td>73.1 91.4 96.0 56.8 81.5 89.2</td>
<td>488.0</td>
<td>94.3 99.4 99.8 82.8 96.7 98.4</td>
<td>571.4</td>
</tr>
<tr>
<td>ALBEF-base+MixGen</td>
<td>3M</td>
<td>72.5 91.7 95.9 55.8 81.3 88.4</td>
<td>485.6</td>
<td>95.1 99.1 99.7 81.4 96.0 98.2</td>
<td>569.5</td>
</tr>
<tr>
<td>ALBEF-base+MixGen</td>
<td>3M</td>
<td>74.2 92.8 96.4 57.3 82.1 89.0</td>
<td>491.8</td>
<td>94.8 99.4 100.0 82.4 96.3 98.0</td>
<td>570.9</td>
</tr>
</tbody>
</table>

Table 2. Zero-shot image-text retrieval on Flickr30K dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Images</th>
<th>Flickr30K (1K test set)</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Text Retrieval</td>
<td>Image Retrieval</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R@1 R@5 R@10 R@1 R@5 R@10</td>
<td>R@1 R@5 R@10 R@1 R@5 R@10</td>
</tr>
<tr>
<td>UNITER-base [3]</td>
<td>4M</td>
<td>80.7 95.7 98.0 66.2 88.4</td>
<td>521.9</td>
</tr>
<tr>
<td>ViLT-base [19]</td>
<td>4M</td>
<td>73.2 93.6 96.5 55.0 82.5</td>
<td>480.3</td>
</tr>
<tr>
<td>ALBEF-base [23]</td>
<td>4M</td>
<td>96.5 98.8 99.9 76.8 93.7</td>
<td>556.2</td>
</tr>
<tr>
<td>ALBEF-base+MixGen</td>
<td>3M</td>
<td>91.1 98.2 99.3 75.7 92.5</td>
<td>552.8</td>
</tr>
<tr>
<td>ALBEF-base+MixGen</td>
<td>3M</td>
<td>91.6 99.2 99.9 77.2 93.6</td>
<td>558.1</td>
</tr>
</tbody>
</table>

Table 3. Compatibility to other vision-language pre-training methods. Adding MixGen leads to consistent performance boost in terms of image-text retrieval. Note the models here are pre-trained on three datasets (COCO, VG and SBU) with 1M images.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Venue</th>
<th>Flickr30K-Ft (1K test set)</th>
<th>MSCOCO-Ft (5K test set)</th>
<th>Flickr30K-Zs (1K test set)</th>
<th>RSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>VILT [19]</td>
<td>ICML 21</td>
<td>232.2 232.2 172.5</td>
<td>135.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLIP + MixGen</td>
<td>ICML 21</td>
<td>241.4 241.4 189.7 189.7</td>
<td>150.4 150.4 149.9 149.9</td>
<td>485.3</td>
<td></td>
</tr>
<tr>
<td>CLIP + MixGen</td>
<td>ICML 21</td>
<td>538.2 538.2 450.5</td>
<td>492.1 492.1 492.1 492.1</td>
<td>557.3</td>
<td></td>
</tr>
<tr>
<td>ALBEF [23]</td>
<td>NeurIPS 21</td>
<td>555.7 555.7 456.3 456.3</td>
<td>518.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALBEF+MixGen</td>
<td>-</td>
<td>561.0 561.0 477.7 477.7</td>
<td>524.3 524.3 524.3 524.3</td>
<td>557.3</td>
<td></td>
</tr>
<tr>
<td>TCL [60]</td>
<td>CVPR 22</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCL + MixGen</td>
<td>-</td>
<td>563.6 563.6 490.2 490.2</td>
<td>539.5 539.5 539.5 539.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2. Image-text retrieval

Image-text retrieval includes two subtasks: (1) retrieve images with given text (Image Retrieval) and (2) retrieve text with given images (Text Retrieval). We conduct experiments on MSCOCO [28] and Flickr30K [36] datasets. In terms of evaluation, we use recall at K (R@K) metric, where K = {1, 5, 10}. For the purpose of easy comparison, we also use RSUM as the metric to reveal the overall performance of models following [52], which is defined as the sum of recall metrics at K = {1, 5, 10} of both image and text retrieval tasks.

Fine-tuned results In Table 1, we can see that MixGen consistently improves over our ALBEF baseline on both datasets. Under the 3M setting, simply adding MixGen without any modifications leads to an improvement of 6.2% RSUM score on COCO and 1.4% RSUM score on Flickr30K. Note that, due to the missing data problem, our reproduced ALBEF (trained on 3.3M pairs) achieves marginally lower performance than the numbers reported in the original paper (trained on 4M pairs, gray row in Table 1). However, after adding MixGen, our performance is even better than the original paper on COCO and competitive on Flickr30K, despite our model is trained with 700K less image-text pairs. This clearly shows that MixGen improves the data efficiency for model training.

Zero-shot results In Table 2, similar conclusions can be ob-

1https://github.com/mlfoundations/open_clip
2https://github.com/dandelion/VILT
3https://github.com/uta-smile/TCL
4https://github.com/salesforce/ALBEF
Table 4. Comparison with state-of-the-art methods on downstream vision-language tasks. MixGen consistently improve across VQA, VR and VE.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Images</th>
<th>VQA test-dev</th>
<th>VQA test-std</th>
<th>NLVR² dev</th>
<th>NLVR² test-P</th>
<th>SNLI-VE val</th>
<th>SNLI-VE test</th>
</tr>
</thead>
<tbody>
<tr>
<td>VisualBERT-base [24]</td>
<td>4M</td>
<td>70.80</td>
<td>71.00</td>
<td>67.40</td>
<td>67.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OSCAR-base [27]</td>
<td>4M</td>
<td>73.16</td>
<td>73.44</td>
<td>78.07</td>
<td>78.36</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UNITER-base [3]</td>
<td>4M</td>
<td>72.70</td>
<td>72.91</td>
<td>77.18</td>
<td>77.85</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ViLT-base [19]</td>
<td>4M</td>
<td>71.26</td>
<td>-</td>
<td>75.70</td>
<td>76.13</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UNIMO-base [26]</td>
<td>4M</td>
<td>73.79</td>
<td>74.02</td>
<td>-</td>
<td>-</td>
<td>80.00</td>
<td>79.10</td>
</tr>
<tr>
<td>VILLA-base [10]</td>
<td>4M</td>
<td>73.59</td>
<td>73.67</td>
<td>78.39</td>
<td>79.30</td>
<td>79.47</td>
<td>79.03</td>
</tr>
<tr>
<td>ALBEF-base [23]</td>
<td>4M</td>
<td>74.54</td>
<td>74.70</td>
<td>80.24</td>
<td>80.50</td>
<td>80.14</td>
<td>80.30</td>
</tr>
<tr>
<td>ALBEF-base [23]</td>
<td>3M</td>
<td>74.38</td>
<td>74.51</td>
<td>-</td>
<td>-</td>
<td>80.23</td>
<td>80.50</td>
</tr>
<tr>
<td>ALBEF-base [23]</td>
<td>3M</td>
<td>74.51</td>
<td>+0.13</td>
<td>74.79</td>
<td>+0.28</td>
<td>80.23</td>
<td>+0.76</td>
</tr>
<tr>
<td>ALBEF [23]</td>
<td>4M</td>
<td>74.54</td>
<td>74.70</td>
<td>80.24</td>
<td>80.50</td>
<td>80.14</td>
<td>80.30</td>
</tr>
<tr>
<td>ALBEF [23]</td>
<td>3M</td>
<td>74.51</td>
<td>+0.13</td>
<td>74.79</td>
<td>+0.28</td>
<td>80.23</td>
<td>+0.76</td>
</tr>
<tr>
<td>ALBEF+MixGen [23]</td>
<td>3M</td>
<td>74.51</td>
<td>+0.13</td>
<td>74.79</td>
<td>+0.28</td>
<td>80.23</td>
<td>+0.76</td>
</tr>
</tbody>
</table>

Figure 3. Image-text retrieval performance given various number of training images. Note that ALBEF with MixGen trained on 3M images achieves competitive performance or outperform baseline ALBEF trained on 4M images. This indicates the data efficiency of MixGen as an effective data augmentation method.

Table 5. Weakly-supervised visual grounding on RefCOCO+

<table>
<thead>
<tr>
<th>Method</th>
<th>#Images</th>
<th>Val</th>
<th>TestA</th>
<th>TestB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARN [29]</td>
<td>14M</td>
<td>52.78</td>
<td>34.35</td>
<td>32.13</td>
</tr>
<tr>
<td>CCL [67]</td>
<td>14M</td>
<td>51.58</td>
<td>36.91</td>
<td>33.56</td>
</tr>
<tr>
<td>ALBEF [23]</td>
<td>14M</td>
<td>58.46</td>
<td>65.89</td>
<td>46.25</td>
</tr>
<tr>
<td>ALBEF+MixGen [23]</td>
<td>3M</td>
<td>57.76</td>
<td>65.08</td>
<td>45.57</td>
</tr>
<tr>
<td>ALBEF+MixGen [23]</td>
<td>3M</td>
<td>56.72</td>
<td>65.35</td>
<td>46.47</td>
</tr>
</tbody>
</table>

Compatibility with other VL models
We show MixGen’s compatibility with other vision-language pre-training methods, i.e., CLIP [38], ViLT [19] and TCL [60]. Besides adding MixGen, we do not modify their original training settings. Given ViLT training is very costly (e.g., 3 days with 64 V100 GPUs), we only use three datasets (COCO, VG, and SBU) instead of four during pre-training for this experiment. The dataset consists of 1M unique images and 2.2M image-text pairs, which we term the 1M setting. As shown in Table 3, simply adding MixGen on top of these strong baselines consistently improve state-of-the-art performance. In terms of fine-tuned image-text retrieval on COCO, MixGen demonstrates significant accuracy boost (absolute): ViLT (+17.2%), CLIP (+4.1%), ALBEF (+7.0%) and TCL (+3.2%). This shows the versatility of MixGen as image-text data augmentation in pre-training.

Data efficiency
We investigate how much data efficiency MixGen can achieve. We reduce the number of unique images used for pre-training, from 3M to 2M, 1M and 200K. We have described 1M and 3M settings. For 2M, we use three datasets plus a random subset from CC dataset. For 200K, we simply use two datasets (COCO and VG). The performance of image-text retrieval can be seen in Figure 3. We first notice that adding MixGen is always better than without it. In particular, the improvement in low data regime is more significant. Secondly, the performance of ALBEF with MixGen when trained on 1M, 2M and 3M samples match baseline ALBEF when trained on 2M, 3M and 4M samples, respectively. This again indicates the data efficiency of MixGen.

3.3. Evaluation on VQA, VR, VE and VG
We first describe the other four downstream tasks. Visual Question Answering (VQA) aims to predict an answer given an image and a corresponding question. We conduct experiments on the VQA 2.0 dataset [11] and consider it as
an answer generation task following [23, 4]. We constrain the fine-tuned answer decoder to generate answers from the set of 3, 192 candidates for fair comparison to other literature. We use the official evaluation server\(^5\) to report accuracy. Visual Entailment (VE) is a visual reasoning task to predict whether the relationship between an image and text is entailment, neutral or contradictory. Following [3, 23], we treat this task as a three-way classification problem and report classification accuracy on the SNLI-VE [53] dataset. Visual Reasoning (VR) requires the model to determine whether a textual statement describes a pair of images. We conduct experiments on the NLVR\(^2\) [43] dataset following [19, 23, 27] and report standard per-example prediction accuracy. Visual Grounding (VG) localizes an image region that corresponds to a text description. Here, we follow [23, 33] to evaluate on the RefCOCO+ dataset [62] in a weakly-supervised setting.

Table 4 reports the performance comparisons of different vision-language pre-training baselines on downstream VQA, VR and VE tasks. Similar to image-text retrieval task, MixGen consistently boosts performance on the three tasks. Under the 3M setting, ALBEF with MixGen outperforms its corresponding baseline by absolute 0.28% on VQA test-std, 0.89% on NLVR\(^2\) test-P, and 0.36% on SNLI-VE test. Note that ALBEF with MixGen under 3M setting even outperforms original ALBEF trained with 4M images on VQA and NLVR\(^2\).

Table 5 reports the performance of visual grounding on RefCOCO+ dataset. ALBEF\(_{itm}\) and ALBEF\(_{itm}^{+}\) are two variants that compute Grad-CAM visualization for grounding. Since ALBEF\(_{itm}\) performs better, we adopt ALBEF\(_{itm}\) as our baseline and add MixGen on top of it. We can see that our model with MixGen trained under 3M setting not only surpasses its corresponding baseline on both test sets, but also outperforms ALBEF\(_{itm}\) trained with 14M dataset on TestB. All empirical results above suggest the superior data efficiency brought by MixGen.

### 3.4. Visualizations of MixGen

#### Image-text retrieval
In Figure 5, we show visualization of text-to-image retrieval on MSCOCO. To be specific, given a text query, we want to compare the rank of the retrieved ground-truth image out of all the retrieved images between ALBEF with and without MixGen. We can see that MixGen is usually able to locate the matching image in top-3 retrievals, performing significantly better than the baseline ALBEF.

**Visual grounding and VQA** In Figure 4, we show Grad-CAM visualizations to help us understand why MixGen is beneficial. For the visual grounding task on the RefCOCO+ dataset, we can see that model trained with MixGen can locate image regions more precisely according to the text query. Even for the failure case, model trained on MixGen is able to attend better on “sitting men”, instead of focusing on the wall (wo MixGen model). For VQA task on VQA2.0 dataset, MixGen can attend to important cues that lead to the correct answer (e.g., black background for predicting if this is a black-white image). For the failure case, telling time might be too challenging, where both ALBEF and ALBEF + MixGen fail.

---

\(^5\)https://visualqa.org/evaluation.html
4. Ablation Studies

In this section, we perform various ablation studies to support the design choice of MixGen. Unless otherwise stated, we adopt ALBEF as baseline and use the 1M setting which consists of three datasets (COCO, VG, and SBU) in model pre-training.

**Design variants** In Sec. 2.1, we introduce 5 variants to support the design choice of MixGen. Detailed comparisons among variants can be found in Table 6. Here we use the same training setting for all 5 variants and evaluate them across five downstream datasets. As we can see in Table 7, our default MixGen achieves the overall best performance, and consistently outperform other variants on different tasks except VQA.

**Input-level vs embedding-level MixGen** Recall in Sec. 2.2 that MixGen can be either performed on the raw input as input-level MixGen, or performed on the embedding level as embedding-level MixGen. Here, we report the overall RSUM scores on COCO and Flicker30K datasets. As we can see in Figure 6 left, both input-level and embedding-level MixGen perform better than the baseline. In addition, input-level MixGen performs consistently better than embedding-level MixGen.

**Optimal ratio of MixGen samples** In our pseudocode, we replace the first $M$ training samples with the newly generated pairs by MixGen, so that the total batch size and training iterations stay the same. Motivated by mixup [65], we could randomly shuffle existing pairs and replace them all by new samples, i.e., $M = B$. Here, we perform an ablation study to investigate how many new pairs is optimal. As we can see in Figure 6 right, given a fixed batch size $B = 512$ while varying $M, M = 128$ (i.e., $M = B/4$) leads to the best performance.

5. Discussion

While we have shown MixGen’s effectiveness across multiple settings, there are many more factors that we do not consider in this paper. First, there are many popular unimodal data augmentation techniques we could compare to [18, 47]. However, due to its unlimited combination, we can only choose the most related configurations for evaluation, and leave the rest comparisons for future work. Second, due to the computational limits, we only perform pre-training on the most widely adopted 4M set and compare to other work on this setting for fair comparison. However, there are larger-scale datasets becoming public available recently that we could explore, such as LAION-400M [41]. Third, despite we have evaluated MixGen on four models and five vision-language tasks, there are more models and tasks in vision-language domain. For example, several recent work [22, 61, 50] have adopted encoder-decoder architecture and reported performance on generation tasks like image captioning. There are also work in video-text retrieval [32], text-to-image generation [39], language guided object detection [25] or segmentation [57]. It would be interesting to investigate the generalization capability of MixGen for these methods and tasks.

### Table 6. MixGen variants

For (d) and (e), $|T|_\lambda$ indicates randomly keeping $\lambda \cdot |T|$ tokens in text sequence $T$, where $|T|$ is the number of tokens in this sequence. See text for more details.

<table>
<thead>
<tr>
<th>Variants</th>
<th>Image</th>
<th>Text</th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MixGen</td>
<td>$I_k = \lambda \cdot I_i + (1 - \lambda) \cdot I_j$</td>
<td>$T_k = \text{concat}(T_i, T_j)$</td>
<td>$\lambda = 0.5$</td>
</tr>
<tr>
<td>(a)</td>
<td>$I_k = \lambda \cdot I_i + (1 - \lambda) \cdot I_j$</td>
<td>$T_k = \text{concat}(T_i, T_j)$</td>
<td>$\lambda \sim \text{Beta}(0.1, 0.1)$</td>
</tr>
<tr>
<td>(b)</td>
<td>$I_k = \lambda \cdot I_i + (1 - \lambda) \cdot I_j$</td>
<td>$T_k = \text{concat}(T_i, T_j)$</td>
<td>$\lambda = 0.5$</td>
</tr>
<tr>
<td>(c)</td>
<td>$I_k = I_i$ or $I_j$</td>
<td>$T_k = \text{concat}(T_i, T_j)$</td>
<td>-</td>
</tr>
<tr>
<td>(d)</td>
<td>$I_k = \lambda \cdot I_i + (1 - \lambda) \cdot I_j$</td>
<td>$T_k = \text{concat}(</td>
<td>T_i</td>
</tr>
<tr>
<td>(e)</td>
<td>$I_k = \lambda \cdot I_i + (1 - \lambda) \cdot I_j$</td>
<td>$T_k = \text{concat}(T_i, T_j)_{0.5}$</td>
<td>$\lambda \sim \text{Beta}(0.1, 0.1)$</td>
</tr>
</tbody>
</table>

### Table 7. Ablation of MixGen design

For image-text retrieval task, we report the RSUM metric. Ft: fine-tuned setting. Zs: zero-shot setting.

<table>
<thead>
<tr>
<th>MixGen variants</th>
<th>Flickr30K-ft (1K test set)</th>
<th>MSCOCO-ft (5K test set)</th>
<th>Flickr30K-Zs (1K test set)</th>
<th>SNLI-VE (test)</th>
<th>NLVR² (test-P)</th>
<th>VQA (test-dev)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALBEF-base</td>
<td>555.7</td>
<td>470.7</td>
<td>518.0</td>
<td>78.91</td>
<td>78.09</td>
<td>73.62</td>
</tr>
<tr>
<td>MixGen</td>
<td>561.0</td>
<td>477.7</td>
<td>524.3</td>
<td>79.65</td>
<td>79.42</td>
<td>73.84</td>
</tr>
<tr>
<td>(a)</td>
<td>553.2</td>
<td>467.4</td>
<td>512.9</td>
<td>78.68</td>
<td>77.22</td>
<td>73.15</td>
</tr>
<tr>
<td>(b)</td>
<td>557.2</td>
<td>470.0</td>
<td>516.2</td>
<td>78.78</td>
<td>78.23</td>
<td>73.76</td>
</tr>
<tr>
<td>(c)</td>
<td>555.4</td>
<td>472.3</td>
<td>518.2</td>
<td>78.87</td>
<td>78.60</td>
<td>73.34</td>
</tr>
<tr>
<td>(d)</td>
<td>555.2</td>
<td>463.7</td>
<td>506.8</td>
<td>79.07</td>
<td>78.30</td>
<td>73.29</td>
</tr>
<tr>
<td>(e)</td>
<td>559.2</td>
<td>477.1</td>
<td>523.7</td>
<td>79.36</td>
<td>78.33</td>
<td>73.95</td>
</tr>
</tbody>
</table>
MixGen is simple, effective, and compatible with generic vision-language pre-training methods.

Relation to mixup Mixup was originally introduced as a data-agnostic augmentation technique for CV [65]. It is a widely adopted (even by-default) technique for training CNNs or vision transformers. Specifically, mixup generates new training samples via

\[ x_k = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \]
\[ y_k = \lambda \cdot y_i + (1 - \lambda) \cdot y_j, \]  

(3)

where \( x_i \) and \( x_j \) are raw input vectors, \( y_i \) and \( y_j \) are one-hot label encodings, and \( \lambda \in [0, 1] \) is sampled from a Beta distribution. Recently, [12] adapted mixup to the NLP domain by replacing raw image pixels \( x \) with text embeddings.

MixGen differs from mixup in several aspects. First, MixGen is proposed for multi-modal data augmentation. The input to MixGen is data from different modalities, and there is no one-hot label encodings \( y \) involved in Equation (1). Second, MixGen does not perform mixup on the text modality. Instead, MixGen simply concatenates text sequences to perform image-text co-augmentation and best preserves information.

7. Conclusion

In this work, we present a new vision-language joint data augmentation method termed MixGen. Adding MixGen on four recent state-of-the-art models achieves consistent improvement across five different downstream tasks. Strong empirical results suggest that MixGen not only makes these models learn better multi-modal latent representations, but also improves their data efficiency. We hope that this work will provide useful data points for future research on joint multi-modal data augmentation.

References


