Appedix for VinVL: Revisiting Visual Representations in Vision-Language Models

A. Qualitative study of three pre-trained vision models

We apply three (pre-trained) object detection models on the image in Figure 1 and list their detection results for a more detailed comparison.

Detections from X152-FPN trained on Open Images V5. See Figure 2:

Surfboard; Surfboard; Surfboard; Surfboard; Man; Human leg; Human leg; Swimwear; Swimwear; Shorts; Shorts; Boy; Human arm.

Detections from R101-C4 trained on VG by Anderson *et al*. [2]. There are obviously wrong detections, marked in red. See Figure 3 (top):

black shorts; young, shirtless, standing, barefoot, surfing, little, playing boy; shirtless, standing, barefoot, walking, wet, surfing, young man; tan, bare, shirtless back; blue, clear, cloudy, hazy, light blue sky; young, shirtless, standing, surfing, barefoot, little boy; brown, short, wet, blond hair; brown, short, wet, blond hair; small, crashing wave; white, wet surfboard; white, crashing, big, rolling wave; wet, tan surfboard; green, blue fin; blue, calm, choppy, wavy, ocean, splashing, foamy, water, rough, sandy, wet ocean; wet, calm, sandy, splashing, wavy water; white, wet surfboard; bare, wet foot; blue, colorful, multi colored, floral shorts; calm, choppy, water, rough, foamy, wavy water; distant, rocky, hazy mountains; standing, shirtless, young, barefoot, wet, surfing, walking, smiling boy; calm ocean; distant, rocky mountain; white, bare, wet surfboard; wet, sandy, calm, tan beach; gray, big rock; blue, calm background; wet, brown, tan, sandy sand; wet shadow; blue, colorful, floral, multi colored swim trunks; yellow, plastic hand.

Detections from our pre-trained X152-C4 model pre-trained on four datasets and fine-tuned on VG. There are some repetitive detections, but no obvious wrong detections. See Figure 3 (bottom):

blue, green fin; young, barefoot, shirtless, standing, surfing, smiling, little, playing, looking, blond boy; young, barefoot, standing, shirtless, smiling, surfing, blond, playing, looking, little, walking, riding boy; shirtless, barefoot, standing, young, smiling, surfing, walking, wet, playing man; bare, wet foot; black, white surfboard; small, large, white, crashing, big, water, rolling, splashing, rough, foamy wave; bare, wet foot; dark, black, wet, cast shadow; blue, clear, hazy, cloudy, cloudless sky; black, gray, white, raised surfboard; black, wet, short short; brown, short, blond, wet, curly, wavy hair; distant, brown, large, rocky, hazy, big mountain; brown, short, dark, blond, wet hair; blue, white, calm, wavy, choppy, ocean, splashing, water, rough, clear, shallow water; bare, tan, light, beige back; black, blue, wet surfboard; small, dark, water, crashing, rolling, splashing, big wave; wet, white, sandy, tan surfboard; blue, colorful, floral, multi colored, patterned trunk; wet, brown, sandy, tan sand; white, blue, calm, foamy, choppy, splashing, wavy, ocean, rough, water, clear, shallow water; wet, brown, sandy, calm, tan, shallow, smooth, muddy, rough beach; black, white, young board; shirtless, young, standing, barefoot, smiling, surfing, looking, walking, playing boy; blue, calm, choppy, wavy, ocean, clear, rough, splashing, water, foamy, shallow, rippled ocean; yellow, gold bracelet; white, silver, black logo; wet, bare, bent, tan, crossed, hairy, short, skinny, back, muscular, extended, outstretched leg; black, gray, white board; brown, distant, large, rocky, big hill; brown, short, blond, wet, curly head; red, black logo; bare, raised, extended, holding, open, up, bent, outstretched hand; black, wet swim trunks; bare, wet, bent, tan, crossed, skinny, short, back, muscular leg; wet, brown, muddy, sandy, tan, shallow reflection.

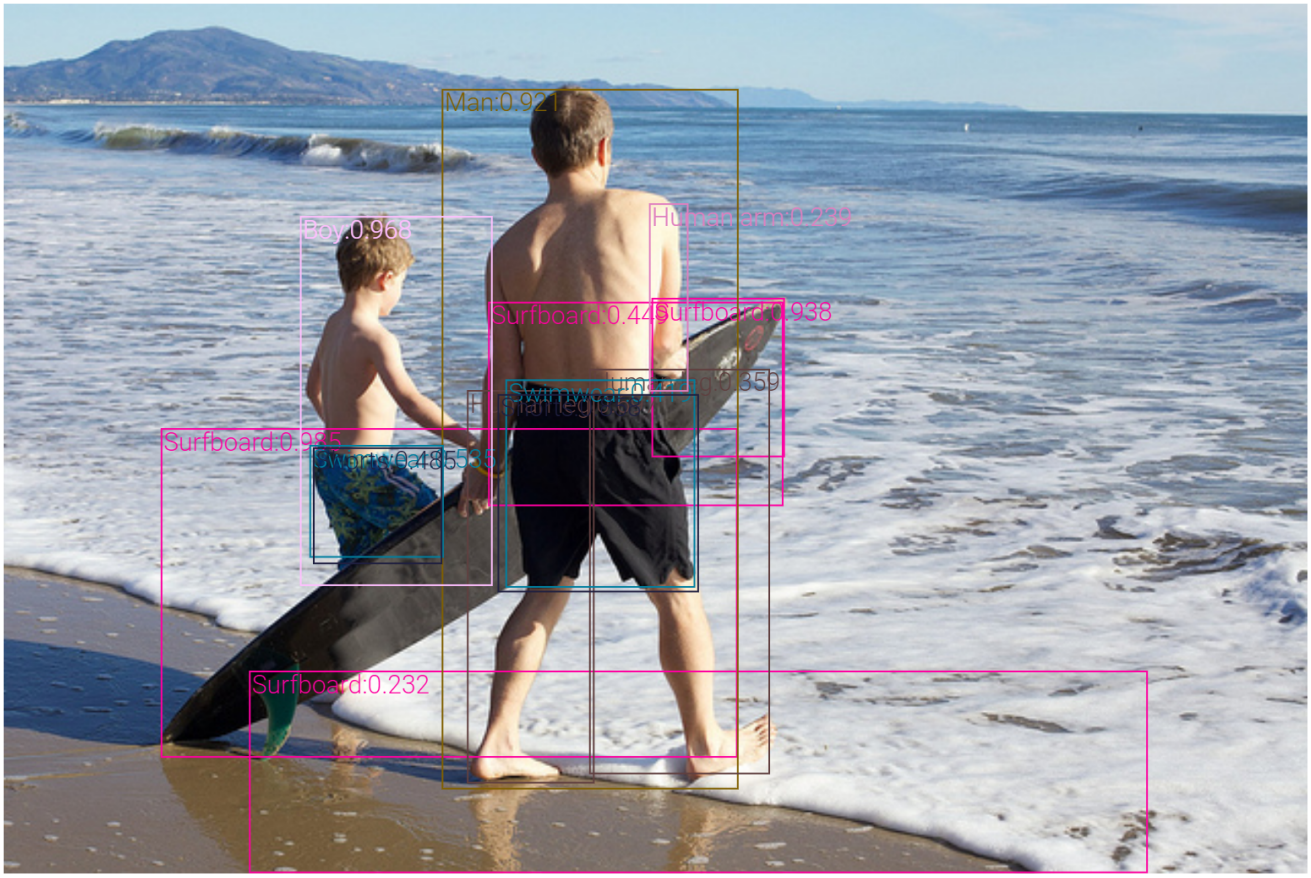

Figure 2: Predictions from X152-FPN trained on OpenImages. Test image: COCO_test2015_000000028839

B. OSCAR+ pre-training

B.1. Pre-training Corpus

Table 17 shows the statistics of image and text of the pre-training corpora. In our ablation study, we use corpora of three different sizes: 'Small', 'Medium', 'Large'. Different from OSCAR [20], we make use of image tagging datasets OpenImages, by generating captions using OSCAR's image captioning model to form triplets of (generated caption, image tags, image features) for OSCAR+ pre-training. By self-training technique, our pre-training corpora can be scaled to a much larger amount by making use of large-scale image tagging datasets, e.g., OpenImages (9M) and YFCC (92M).

B.2. OSCAR+ pre-training objectives

Masked Token Loss: A Loss Mimics Image Captioning. The word tokens of image captions (questions) $\boldsymbol{w}$ and word tokens of object tags (answers) $\boldsymbol{q}$ share the same linguistic semantic space, and the Masked Token Loss (MTL) is applied on tokens of both $\boldsymbol{w}$ and $\boldsymbol{q}$. We define the *discrete token sequence* as $\boldsymbol{h} \triangleq [\boldsymbol{w}, \boldsymbol{q}]$, and apply the Masked Token Loss (MTL) for pre-training. At each iteration, we randomly mask each input token in $\boldsymbol{h}$ with probability 15%, and replace the masked one h_i with a special token [MASK]. The goal of training is to predict these masked tokens based on their surrounding tokens $\boldsymbol{h}_{\setminus i}$ and image features $\boldsymbol{v}$ by minimizing the negative log-likelihood:

$$\mathcal{L}_{\text{MTL}} = -\mathbb{E}_{(\boldsymbol{v},\boldsymbol{h})\sim\mathcal{D}} \log p(h_i|\boldsymbol{h}_{\setminus i}, \boldsymbol{v}) \quad (5)$$

This is the same MTL as in OSCAR [20] and similar to the masked language model used by BERT. The masked word or tag needs to be recovered from its surrounding context, with additional image information to help ground the learned word embeddings in the vision context.

3-way Contrastive Loss: A Loss Mimics Text-Image Retrieval and Visual Question Answering Simultaneously. We present our 3-way contrastive loss in Section 3.2 in the main paper.

B.3. Ablation of the two new techniques

Effect of self-training: Leveraging Image Tagging data. In Figure 4, we show the effect of self-training by making use of tagging data in OSCAR+, by fine-tuning OS-

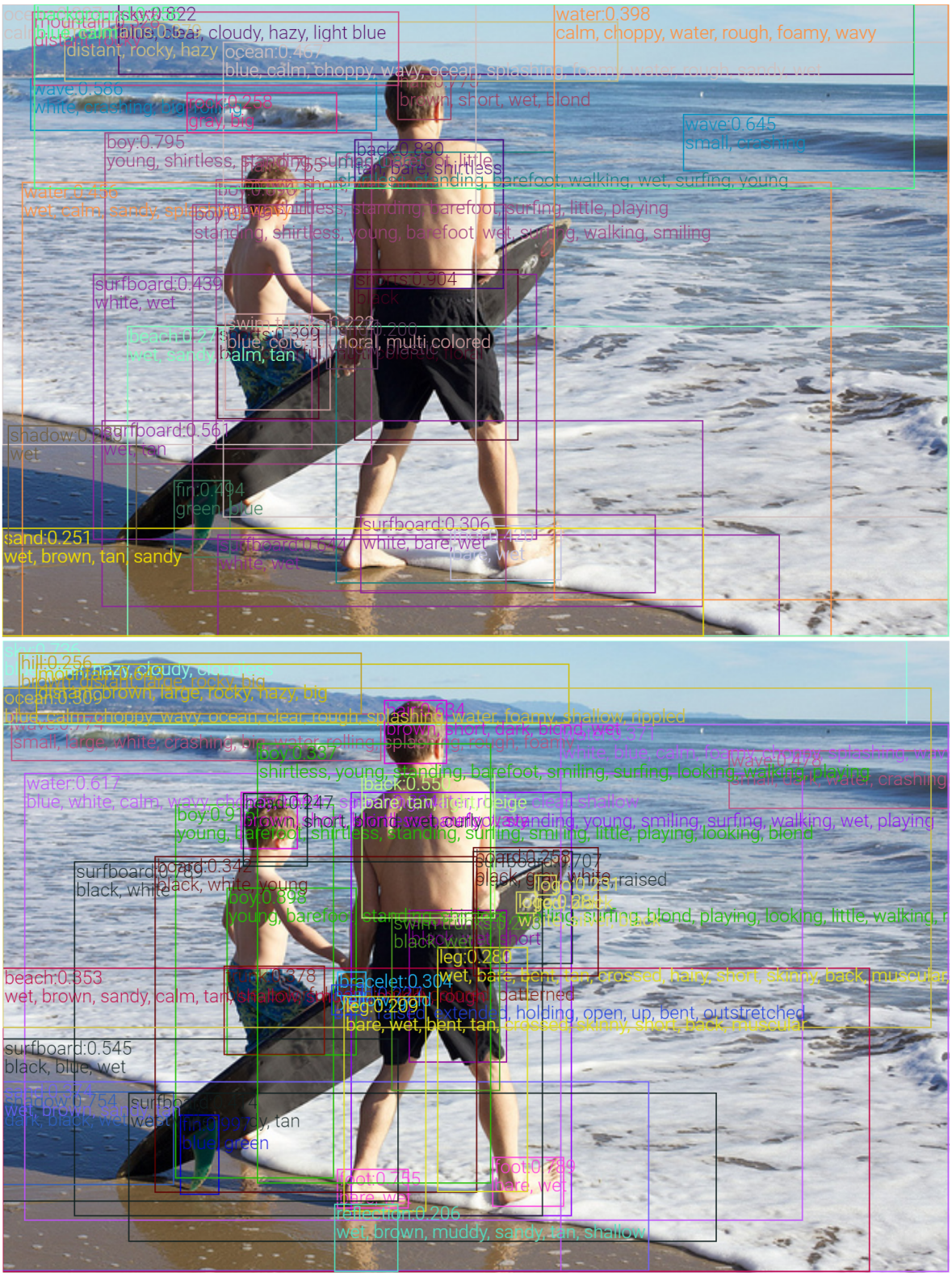

Figure 3: Predictions from R101-C4 trained on VG from [2] (top), X152-C4 pre-trained on 4 OD datasets and finetuned on VG (bottom). Test image: COCO_test2015_000000028839

Small	0.22M Images, 2.5M QAs, 0.7M captions							
Medium	1.89M Images, 2.5M QAs, 0.7M captions, 1.67M pseudo-captions							
Large	5.65M Images, 2.5M QAs, 4.68M captions, 1.67M pseudo-captions							
Source	VQA (train)	GQA (bal-train)	VG-QA (train)	COCO (train)	Flicker30k (train)	OpenImages (od train)	CC (train)	SBU (all)
Image/Text	83k/545k	79k/1026k	87k/931k	112k/559k	29k/145k	1.67M/1.67M	3.1M/3.1M	875k/875k
$\boldsymbol{w}, \boldsymbol{q}, \boldsymbol{v}$	Question, Answer, ImageFeatures			(Generated) Caption, (Generated) ImageTags, ImageFeatures				

Table 17: Statistics of the pre-training corpus.

CAR+ pre-training checkpoints on VQA. Compared with "OSCAR+, Small; VinVL" (green), "OSCAR+, Medium; VinVL" (yellow) adds the 1.7M OpenImages Tagging data into pre-training and its performance gets improved significantly, demonstrating the effect of self-training by making use of tagging data. As baselines, we also provide performance of OSCAR and OSCAR+ with image features from [2], which clearly demonstrates that the new image features pre-trained by VinVL matter significantly in the VL pre-training and VL downstream tasks.

Effect of the new 3-way contrastive loss. As illustrated in Table 3, with the new 3-way contrastive loss, the VQA performance is the same as the OSCAR pre-training, while the Text-Image Retrieval performance improves significantly compared with the OSCAR pre-training.

Overall improvement from OSCAR to OSCAR+. We point out that the improvement from OSCAR to OSCAR+ with image features from [2] is minor, because (1) we only add 1.7M OpenImages' tagging data to enlarge the pre-training corpus, which is a small portion compared with OSCAR's original pre-training corpus (i.e., Large\OI, 3.98M images and 7.18M image-caption pairs), and (2) the new 3-way contrastive loss has more significant improvements in Text-Image Retrieval tasks than that in the VQA task, as illustrated in Table 3. We would expect much more significant improvements when we scale up the OSCAR+'s pre-training corpus to a much larger scale by adding large scale image tagging datasets, e.g., OpenImages (9M) and YFCC (92M).

C. Downstream Tasks Fine-tuning

We follow the downstream task fine-tuning recipes in OSCAR [20].

C.1. VQA

Given an image and a question, the task is to select the correct answer from a multi-choice list, it requires the model to answer natural language questions based on an image. Here we conduct experiments on the widely-used VQA v2.0 dataset [8], which is built on the MSCOCO [24] images. Following [2], for each question, the model picks the corresponding answer from a shared set of $3,129$ candidates.

When fine-tuning on the VQA task, the input sequence contains the concatenation of a given question, object tags and object region features, and then the [CLS] output from OSCAR+ is fed to a task-specific linear classifier for answer prediction. Similarly as the literature [2], we treat VQA as a multi-label classification problem – assigning a soft target score to each answer based on its relevancy to the human answer responses, and then we fine-tune the model by minimizing the cross-entropy loss computed using the predicted scores and the soft target scores. During inference, we simply use Softmax for answer prediction.

For VQA training, we random sample a set of 2k images from the MS COCO validation set as our validation set, the rest of images in the training and validation are used in the VQA fine-tuning. For the OSCAR+$_{\text{B}}$ model, we fine-tune for 25 epochs with a learning rate of $5e^{-5}$ and a batch size of 128. For the OSCAR+$_{\text{L}}$ model, we fine-tune for 25 epochs with a learning rate of $3e^{-5}$ and a batch size of 96.

C.2. GQA

Similarly as VQA, GQA tests the reasoning capability of the model to answer a question. We conduct experiments on the public GQA dataset [12]. For each question, the model chooses an answer from a shared set of $1,852$ candidates. Our fine-tuning procedure is following Oscar [20, 3], which first fine-tunes the model on unbalanced "all-split" for 5 epochs with a learning rate of $5e^{-5}$ and a batch size of 128, and then fine-tuned on the "balanced-split" for 2 epochs.

C.3. Image Captioning

An image captioning model generates a natural language description for a given image. To enable sentence generation, we fine-tune OSCAR+ using the seq2seq objective.

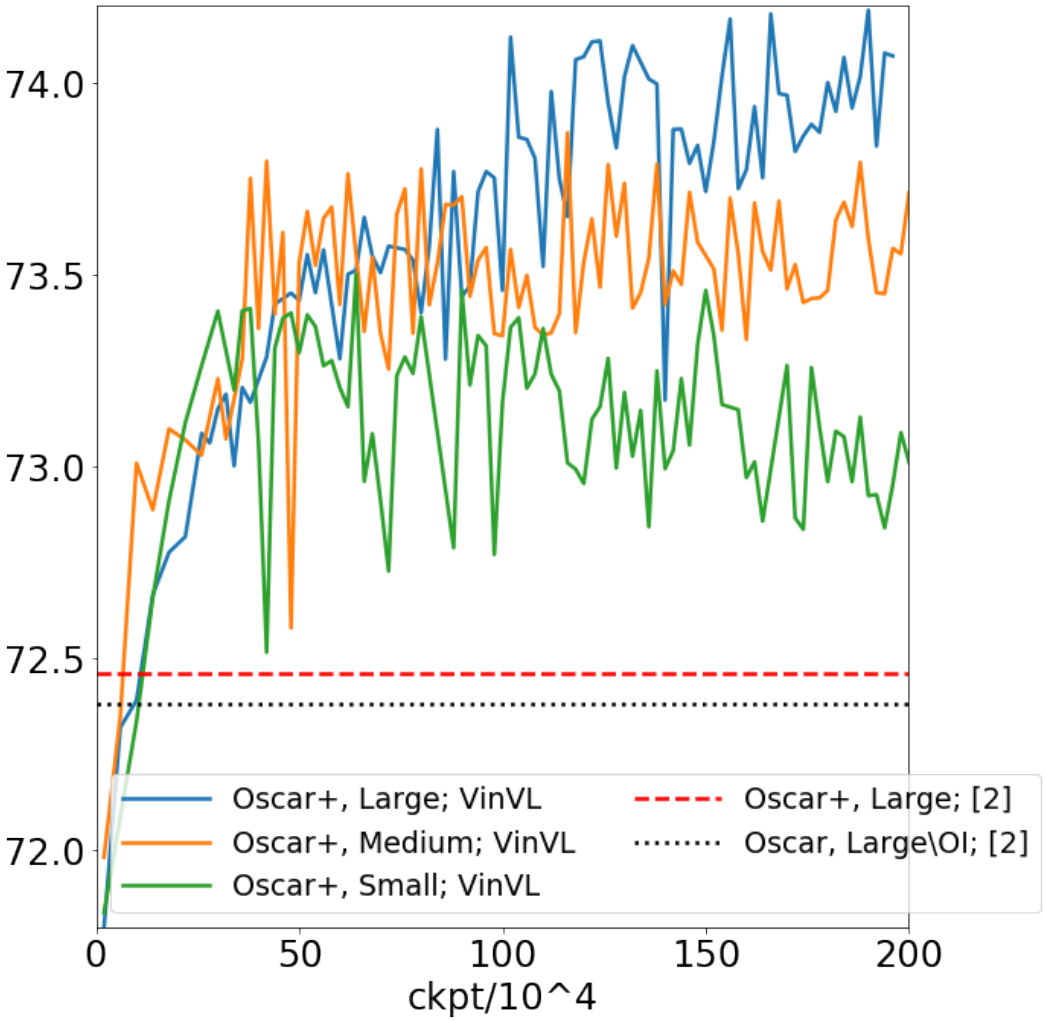

Figure 4: Effect of OSCAR+ pre-training corpus size and effect of self-training by making use of tagging data in OSCAR+. Each curve, with legend "VLP, Corpus; VisionFeature", denotes a VLP experiment where the VLP method is either OSCAR or OSCAR+, the VLP pre-training Corpus is Small/Medium/Large (defined in Table 17), and VisionFeature is either our new vision features (VinVL for short) or those from [2] ([2] for short). X-axis denotes the pre-training iterations of OSCAR+ checkpoints. Y-axix is the vqa-dev accuracy of a VQA model initialized from the corresponding pre-training checkpoint and fine-tuned with a fixed scheme. Compared with "OSCAR+, Small; VinVL" (green), "OSCAR+, Medium; VinVL" (yellow) adds the 1.7M OpenImages Tagging data into the pre-training and its performance gets improved significantly, demonstrating the effect of self-training by making use of tagging data. The "OSCAR+, Large; VinVL" (blue) further scales up the pre-training corpus by adding Google Conceptual Captions and SBU datasets with generated tags and its performance gets further improved, demonstrating the effect of OSCAR+ pre-training corpus size. As baselines, we also provide performance of OSCAR and OSCAR+ with image features from [2], which clearly demonstrates that our new image features (VinVL) matter significantly in the VL pre-training and VL downstream tasks.

The input samples are processed to triples consisting of image region features, captions, and object tags, in the same way as that during the pre-training. We randomly mask out 15% of the caption tokens and use the corresponding output representations to perform classification to predict the token ids. Similar to previous works [20, 44], the self-attention mask is constrained such that a caption token can only attend to the tokens before its position to simulate a uni-directional generation process. Note that all caption tokens will have full attentions to image regions and object tags but not the other way around.

During inference, we first encode the image regions, ob-

ject tags, and a special token [CLS] as input. Then the model starts the generation by feeding in a [MASK] token and selecting a token from the vocabulary based on the likelihood output. Next, the [MASK] token in the previous input sequence is replaced with the selected token and a new [MASK] is appended for the next word prediction. The generation process terminates when the model outputs the [SEP] token. We use beam search (*i.e.* beam size = 5) [2] in our experiments and report our results on the COCO image captioning dataset.

Though the training objective (*i.e.* seq2seq) for image captioning is different from that used in pre-training (*i.e.* bidirectional attention-based masked token loss), we directly fine-tune OSCAR+ for image captioning on COCO without additional pre-training on Conceptual Captions [31]. This is to validate the generalization ability of the OSCAR+ models for generation tasks. We use the same Karpathy split [14]. For the OSCAR+$_{\text{B}}$ model, we fine-tune with cross-entropy loss for 30 epochs with a batch size of 256 and an initial learning rate of $1e^{-5}$ and then with CIDEr optimization [29] for 10 epochs with a batch size of 128 and initial learning rate of $2e^{-6}$. We compare with several existing methods, including BUTD [2], VLP [44], AoANet [10], OSCAR [20].

C.4. NoCaps

Novel Object Captioning [1] extends the image captioning task, is to test models' capability of describing novel objects from the Open Images dataset [16] which are not seen in the training corpus. Following the restriction guideline of NoCaps, we train OSCAR+ on COCO without the initialization from pre-training, so no additional image-text pairs are used for training except COCO.

Since NoCaps images are collected from Open Images, we train an object detector using the Open Images training set and apply it to generate the tags. We conduct experiments from BERT model directly without pre-training as required by the task guidelines. For the OSCAR+$_{\text{B}}$ model, we train 30 epochs with a batch size of 256 and learning rate $1e^{-4}$; further we perform CIDEr optimization with learning rate $5e^{-6}$ and batch size 112 for 10 epochs. During inference, we use constrained beam search for decoding. We compare OSCAR+ with OSCAR [20] on this task.

C.5. Image-Text Retrieval

There are two sub-tasks: *image retrieval* and *text retrieval*, depending on which modality is used as the retrieved target. Both tasks calculate a similarity score between an image and a sentence, which heavily relies on the cross-modal representations.

Following Oscar [20], we formulate the retrieval as a binary classification problem, where given an aligned image-text pair, we randomly select a different image or a different sentence to form an unaligned pair. The final representation of [CLS] is used as the input to the classifier to predict whether the given pair is aligned or not. In the testing stage, the probability score is used to rank the given image-text pairs of a query.

Following [18], we report the top-K retrieval results on both the 1K and 5K COCO test sets. We adopt the widely used Karpathy split [14] on the COCO caption dataset [24] to conduct our experiments. Specifically, the dataset consists of $113,287$ images for training, $5,000$ images for validation, and $5,000$ images for testing. Each image is associated with 5 human-generated captions. For the OSCAR+$_{\text{B}}$ model, we fine-tune with a batch size of 256 for 40 epochs. The initial learning rate is set to $2e^{-5}$ and linearly decreases. For the OSCAR+$_{\text{L}}$ model, we fine-tune with a batch size of 128 for 40 epochs. The initial learning rate is set to $1e^{-5}$ and linearly decreases. We use the validation set for parameter tuning. We compare with several existing methods, including DVSA [14], VSE++ [5], DPC [43], CAMP [38], SCAN [17], SCG [32], PFAN [37], Unicoder-VL [18], 12-in-1 [26], UNITER [4].

C.6. NLVR2

Given a pair of images and a natural language, the goal of NLVR2 [34] is to determine whether the natural language statement is true about the image pair. For NLVR2 fine-tuning, we first construct two input sequences, each containing the concatenation of the given sentence (the natural language description) and one image, and then two [CLS] outputs from OSCAR+ are concatenated as the joint input for a binary classifier, implemented by an MLP.

For the OSCAR+$_{\text{B}}$ model, we fine-tune for 20 epochs with learning rate $\{2e^{-5}, 3e^{-5}, 5e^{-5}\}$ and a batch size of 72. For the OSCAR+$_{\text{L}}$ model, we fine-tune for 20 epochs with learning rate of $\{2e^{-5}, 3e^{-5}\}$ and a batch size of 48.

D. More on the Effect of the Object-Attribute Vocabulary Size: disentangling the effects of region proposals and model weights

In Section 4.2, we demonstrate that the more diverse the visual concepts (object and attribute vocabularies) are, the better the visual region features for VL tasks. The better performance may come from the more diverse proposed regions where the region features are extracted (see the comparison in Figure 1, "region" for short), or from the better model weights that can produce better high-dimensional region representation even for the same region ("model" for short). In this section, we disentangle effects of region proposals and model weights, by performing synthetic experiments in which we use region proposals from one vision model and model weights from another vision model. Our results show that both the region proposals and model

Method	in-domain CIDEr	in-domain SPICE	near-domain CIDEr	near-domain SPICE	out-of-domain CIDEr	out-of-domain SPICE	overall CIDEr	overall SPICE	in-domain CIDEr	in-domain SPICE	near-domain CIDEr	near-domain SPICE	out-of-domain CIDEr	out-of-domain SPICE	overall CIDEr	overall SPICE
	Validation Set								Test Set							
UpDown$^+$	79.3	12.4	73.8	11.4	71.7	9.9	74.3	11.2	76.0	11.8	74.2	11.5	66.7	9.7	73.1	11.2
OSCAR$_B$*	83.4	12.0	81.6	12.0	77.6	10.6	81.1	11.7	81.3	11.9	79.6	11.9	73.6	10.6	78.8	11.7
OSCAR$_L$*	85.4	11.9	84.0	11.7	80.3	10.0	83.4	11.4	84.8	12.1	82.1	11.5	73.8	9.7	80.9	11.3
Human [1]	84.4	**14.3**	85.0	**14.3**	**95.7**	**14.0**	87.1	**14.2**	80.6	**15.0**	84.6	**14.7**	**91.6**	**14.2**	85.3	**14.6**
VIVO* [9]	92.2	12.9	87.8	12.6	**87.5**	11.5	88.3	12.4	89.0	12.9	87.8	12.6	**80.1**	11.1	**86.6**	12.4
VinVL*	**96.8**	13.5	**90.7**	13.1	87.4	11.6	**90.9**	12.8	**93.8**	13.3	**89.0**	12.8	66.1	10.9	85.5	12.5
VinVL+VIVO	**103.7**	**13.7**	**95.6**	**13.4**	83.8	**11.9**	**94.3**	**13.1**	**98.0**	**13.6**	**95.2**	**13.4**	78.0	**11.5**	**92.5**	**13.1**

Table 18: NoCaps evaluation results. All the models are trained on COCO without additional image-caption pairs following the restriction of NoCaps. (UpDown$^+$ is UpDown+ELMo+CBS, the models with * is +SCST+CBS, VinVL+VIVO is with SCST only.)

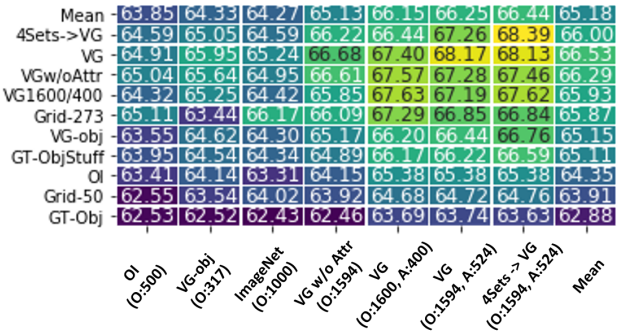

Figure 5: Overall comparison of vocabulary effect on VQA. X-axis: how the R50-C4 model is trained; Y-axis: how the feature is extracted (grid or region features, different kinds of boxes to extract region features). All region features have maximal 50 regions. The top row "Mean" is the average over all rows, showing the overall quality of different vision models. The far-right column "Mean" is the average over all columns, showing the overall quality of different feature extraction methods.

weights matter for VL tasks.

D.1. Disentangling the effects of region proposals and model weights on R50-C4

As in Section 4.2, We train vision models $\boldsymbol{v} = \textbf{Vision}(Img)$ on different datasets, i.e., OpenImages with 500 object classes (OI:O500), standard ImageNet with 1K classes (ImageNet:O1000), Visual Genome with 317 object classes (VG-obj), Visual Genome with 1594 object classes (VG:O1594), VG with 1594 object classes and 524 attribute classes (VG:O1594A524), pretrain on the merged 4 datasets and finetune on VG:O1594A524 (4Sets→VG:O1594A524). For *each* model, we also try different ways to extract features: (1) *region* features from different models' proposed regions (same notations with models) where each image has maximal 50 region features, and (2) *grid* features where we use all grid features (Grid-273) or randomly sampled 50 grid features (Grid-50) for each image. We present the results of these model-region cross-combination experiments in Figure 5. We also present the mean accuracy over all box types to obtain a robust ranking of different checkpoints and the mean accuracy over all checkpoints to obtain a robust ranking of different box types. We have the following observations:

- The richer the object vocabulary is, the better for VQA: OI:500 $\approx$ VG-obj:O317 $<$ ImageNet:O1000 $<$ VG:O1594.
- Attribute information is crucial to VL tasks: all features trained with attributes (Columns with VG:O1594A524) are significantly better than those without attributes.
- Even for small vision backbone R50, vision pre-training makes vision features better: Column "4Sets→VG:O1594A524" are better than all other columns. Notice that the vision pre-training improves both the region features and the grid features.
- It is crucial to extract features from semantically diverse regions: regions from OI and VG-obj are significantly worse than all other regions, and is even worse than grid features.
- Grid features perform worse than region features with regions proposed by VG models. By comparing Row "Grid-273" with rows with VG regions, it seems hopeful to close this gap while paying more hardware memory and computational cost in cross-modal models **VL**. It is three times slower to **train** the "Grid-273" models than training models with region features.

In Figure 6, instead of just showing one final number, we provide the *mean evaluation curves along training trajectories* to demonstrate the ranking, as an even more robust evidence. These results further confirm the conclusions we draw in Section 4.2.

D.2. Disentangling the effects of region proposals and model weights on the SoTA model

In Table 19, we alternate the combination of region proposals and model weights, and evaluate them on VQA. As we can see, the improvement of using boxes from the R101-C4 model [2] to extract features from our X152-

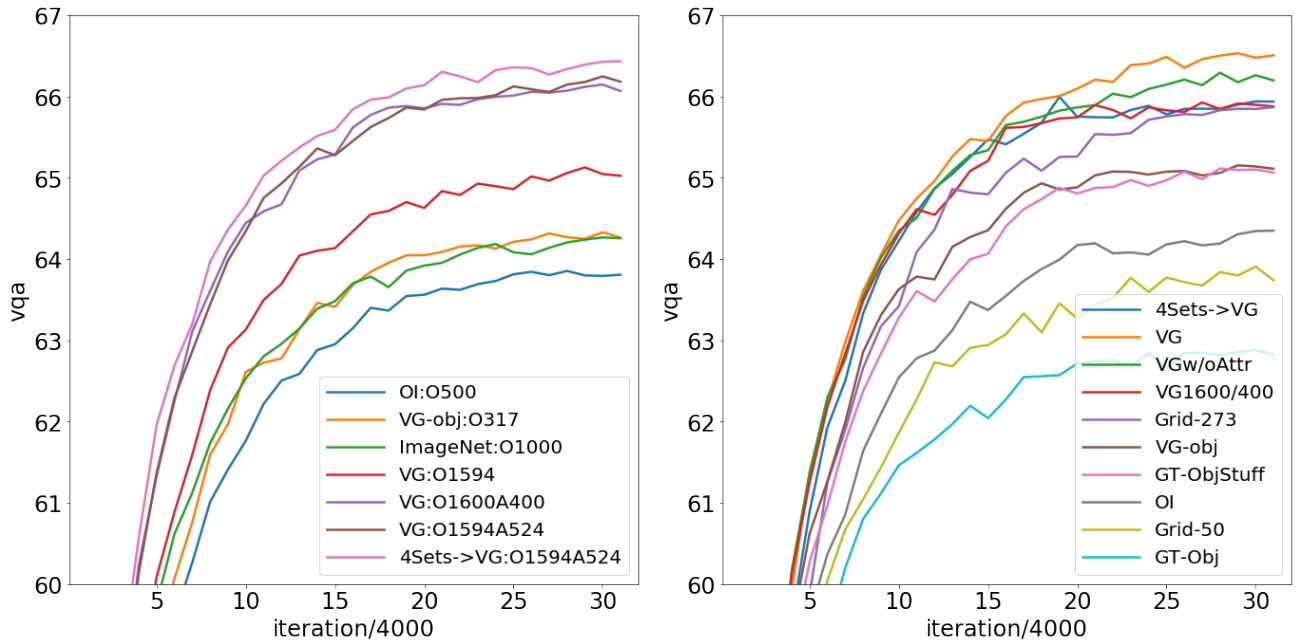

Figure 6: Left: comparison of object vocab and attribute vocab, average over all types of bounding boxes. Right: comparison of feature extraction methods, average over all types of pre-trained vision models. X-axis is the number of iterations when we take the checkpoint for evaluation. Y-axis is the VQA accuracy on our vqa-dev.

C4 model is much bigger than that of using boxes from our X152-C4 model to extract features from the R101-C4 model [2], indicating pre-trained model weights are more important than regions. Inspired by this analysis, we propose the class-agnostic NMS for region selection in the box head of the OD model, which does not sacrifice any VQA performance but greatly improves the model's inference speed. This analysis also suggests that large-scale OD pre-training should improve performance for grid-feature based VL models, as supported by more results in Appendix F.

In Table 19, We also report VQA results with COCO groundtruth object regions (GT-Obj, 80 classes) and object-stuff regions (GT-Obj&Stuff, 171 classes). For VQA task, COCO GT boxes are much worse than proposals from VG trained models. This shows the difference between typical OD tasks and OD in VL: OD in VL requires much richer visual semantics to align with the rich semantics in the language modality. This further echoes with our claim that an image understanding module trained with rich semantics is crucial for VL tasks.

model \ region	GT-Obj	GT-Obj&Stuff	Anderson et al. [2]	VinVL (ours)
Anderson et al. [2]	63.81 ±0.94	66.68 ±0.16	68.52 ±0.11	69.05 ±0.06
VinVL (ours)	65.60 ±0.21	68.13 ±0.26	70.25 ±0.05	71.34 ±0.17

Table 19: Ablation of region and model on VQA.

E. More on FPN and Comparison of C4 and FPN

E.1. Two reasons why FPN performs worse than C4 on VL tasks.

Our experimental results confirm the conclusion of [13] that the FPN model does not provide better region features for VL tasks than the C4 model (Columns "R50C4" vs. "R50FPN" in Table 20). Our analysis reveals two reasons. First of all, all layers involved in feature extraction in the C4 model have been pre-trained using ImageNet while the MLP head of FPN does not. It turns out that the VG dataset is still small to train a good visual features for VL tasks and using ImageNet-pre-trained weights is beneficial. This can be verified by two experiments: (1) When the R50-C4 model is trained on VG with its box head randomly initialized (VG-trained - R50C4 w/ box head randomly initialized), the C4 model's performance is the same as FPN; and (2) C4 and FPN achieve the same performance after vision pre-training on 4 datasets (68.3 vs. 68.2). The second reason is due the network architecture (CNN vs. MLP) of the box head in the OD model. The convolutional head in C4 has a better inductive bias in encoding visual information than the MLP head in FPN. This can be verified by the fact that when vision features from randomly initialized models are used (Row "Initial" in Table 20), R50-C4 performs much better than R50-FPN, indicating that the initial C4

	no image feature $\boldsymbol{w}$	R50-C4 w/ box head randomly initialized	R50-FPN	R50-C4	4Sets→ R50-FPN	4Sets→R50-C4
VG-trained	–	67.6 ±0.13	67.6±0.30	68.0±0.16	68.3±0.11	68.2±0.05
Initial	55.5±0.50	61.8 ±0.47	57.6±0.16	64.8±0.44	66.1±0.23	66.8±0.21

Table 20: C4 vs FPN architecture on VQA. Boxes used to extract features $\boldsymbol{v}$ and tags $\boldsymbol{q}$ used in $\mathbf{VL}$ model are the same with those used in OSCAR [20]. Row "Initial" means using the initialization model without VG training for feature extraction.

features encode much more useful visual information than the inital FPN features. The "random" C4 features nearly match the feature from ImageNet pre-trained model (Row "Initial" Column "R50C4"), while "random" FPN features are close to the performance without visual features as input (Row "Initial" Column "no image feature $\boldsymbol{w}$").

E.2. Effect of pooling methods in FPN on VQA performance.

Different from C4 models that extract region features from a single scale (the end of C4 block), FPN models extract region features from multiple scales adaptively based on the area of the region. Therefore, there is some inhomogeneity in FPN's region features since they may come from different scales. In Figure 7, we show that this is not the cause of FPN's worse performance than C4 on the VQA task. More specifically, we experiment with 4 pooling methods for FPN architecture. (1) adapt: the original FPN's pooling method that extract features adaptively from different scales; (2) max: extract features from all scales and then do a max-pool; (3) avg: extract features from all scales and then do an average-pool; (4) concat: extract features from all scales and then concatenate them together. We also train multiple FPN models on VG with these pooling methods, with or without pre-training on the Objects365 dataset. We experiment on all possible combinations (in total 8×4) of 8 vision models and 4 pooling methods on the VQA task. When there is a parameter dimension mis-match, e.g., non-concat FPN models but use concat pooling methods in VQA and vice versa, we specify those parameter randomly with PyTorch's default initialization method. The results in Figure 7 shows that (1) there is no obvious difference in different pooling methods, with the default "adapt" and the "concat" methods perform slightly better than "max" and "avg"; (2) (without surprise) the performance is significantly worse when there is a parameter dimension mismatch between vision models and VL task feature extraction methods, i.e., non-concat FPN models but use concat pooling methods in VQA and vice versa. These results show that the pooling method (no matter in vision model training or in VL task feature extraction) is not the root cause of FPN's worse performance than C4 on the VQA task.

E.3. Large-scale object-detection pre-training of C4 and FPN models

In this paper, we have trained R50-C4, R50-FPN, R152-C4 and R152-FPN models on the merged object detection datasets described in Table 2. In Figure 8, we report the mAP^{50} of checkpoints from these 4 experiments on 4 validation sets: COCO with stuff (top left), Objects365 (top right), OpenImages (bottom left) and Visual Genome (1594 object classes, bottom right). For R50 models, the R50-FPN model is slightly better than C4 on COCO and Objects365 but slightly worse than C4 on Visual Genome. For R152 models, the R152-FPN model is consistently worse than the R152-C4 model on all 4 different datasets. Therefore, we finally use the R152-C4 model for downstream vision-language tasks.

F. Grid feature

In Table 21, we train grid-feature based and region-feature based X152 models for VQA, with the vision models pre-trained on different vision datasets, i.e., "ImageNet-5k" from [39], our 4-dataset merged OD dataset 2 (4Sets), our VG dataset with 1594 object classes and 524 attribute classes (VG with Attr), and first 4Sets and then VG (4Sets→VG). Vision models in the last three cases are trained with initialization from the same ImageNet-5k checkpoint from [39]. All the region features are extracted with boxes proposed by our best X152-C4 model (pre-trained on 4Sets and fine-tuned on VG). By comparing "ImageNet-5k" and "4Sets→VG", we see that our proposed vision pre-training improves performance for both the grid-feature based model and the region-feature based model. Since the X152 backbone is much larger than the R50 backbone in Figure 5, the larger model makes better use of the large pre-training datasets and thus has more significant improvements. It is interesting to see that for grid-feature based models, the "ImageNet-5k" model performs better than the "4Sets" model and the "VG with Attr", while it is not the case for region-feature based models. This may indicate that how the vision model is trained (grid-feature wise or region-feature wise) may have big impact on the downstream VL tasks.

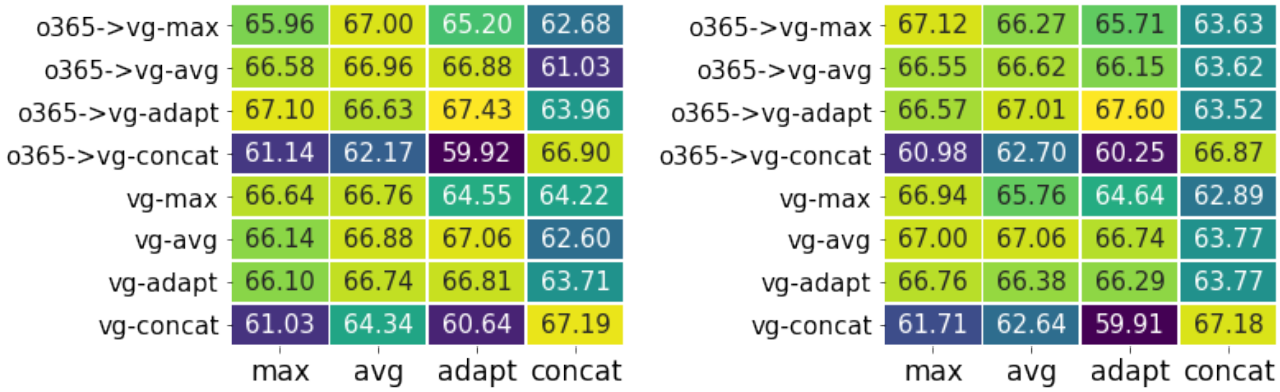

Figure 7: Pooling methods in FPN feature extraction are not the root cause of FPN's worse performance than C4. X-axis: the pooling method when extracting features for VL tasks; Y-axis: the pooling method (vision model) when pre-training the visual feature extraction model. All experiments are using regions from the Bottum-up Top-down model [2]. Each combination is experimented twice with two random seeds, i.e. seed=42 on the left and seed=88 on the right. The results from two random seeds are consistent.

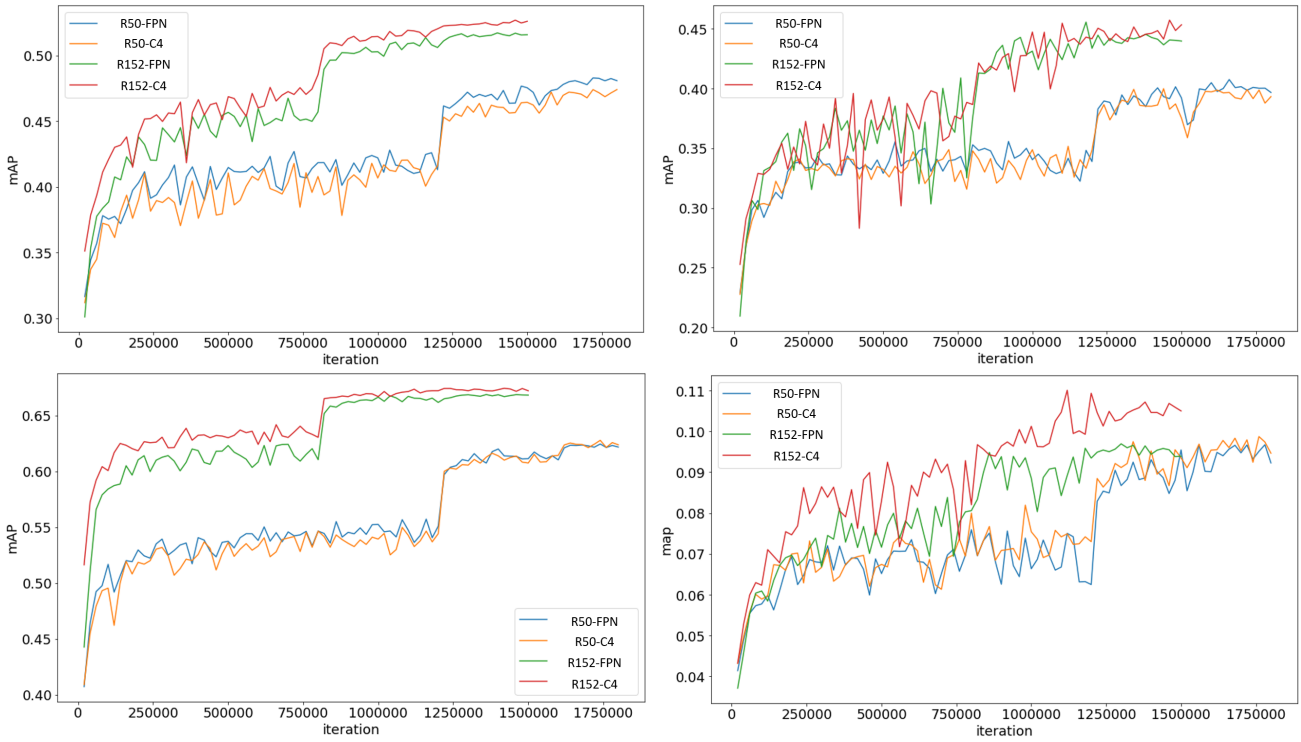

Figure 8: Checkpoints' mAP^{50} on 4 validation sets: COCO with stuff (top left), Objects365 (top right), OpenImages (bottom left) and Visual Genome (1594 object classes, bottom right). For R50 models, the R50-FPN model is slightly better than C4 on COCO and Objects365 but slightly worse than C4 on Visual Genome. For R152 models, the R152-FPN model is consistently worse than the R152-C4 model on all 4 different datasets.

G. End-to-end inference efficiency

We report the end-to-end inference time of different VQA models on a Titan-X GPU and a Xeon E5 CPU in Table 22. For CPU evaluation, we force that the inference use only one CPU thread. The input image size is 800×1333, and we run the inference with batch size 1 (one image-question pair per batch). We can see that (1)

	ImageNet-5k [39]	4Sets	VG with Attr	4Sets→VG
grid feature (273)	68.3±0.29	65.2±2.47	67.5±0.20	69.4*
region feature (50)	67.7±0.16	68.5±0.13	69.8±0.23	70.6±0.13

* The other run failed and thus there is no std for this experiment.

Table 21: Ablation study of X152 models on VQA. Vision models in the last three columns are trained with initialization from the ImageNet-5k checkpoint in the first column. All the region features are extracted with boxes proposed by our best X152-C4 model (pre-trained on 4Sets and fine-tuned on VG). By comparing the first column and the last column, we see that our proposed vision pre-training (first on 4 sets and then on VG with attributes) improves performance for both the grid-feature based model and the region-feature based model. Since the X152 backbone is much larger than the R50 backbone in Figure 5, the larger model can make better use of the large pre-training datasets and thus have more significant improvements.

Model	R50-C4		R101-C4 [2]		X152-C4	
	Vision	**VL**	**Vision**	**VL**	**Vision**	**VL**
Grid-50	0.059±0.018	0.029±0.002	0.083±0.025	0.030±0.003	0.355±0.022	0.031±0.003
Grid-273	0.056±0.005	0.027±0.002	0.082±0.022	0.034±0.001	0.344±0.036	0.037±0.004
Object	0.373±0.040	0.031±0.005	0.663±0.042	0.034±0.003	0.687±0.064	0.036±0.005
Object-eff	0.165±0.029	0.029±0.002	0.442±0.119	0.036±0.003	0.475±0.049	0.037±0.005
Grid-50 (cpu)	1.943±0.244	0.480±0.042	4.050±0.398	0.469±0.046	17.765±1.693	0.501±0.047
Grid-273 (cpu)	2.032±0.230	1.368±0.056	4.052±0.372	1.283±0.067	17.664±1.713	1.326±0.053
Object (cpu)	11.808±1.322	0.500±0.045	31.863±7.932	0.585±0.044	29.641±3.097	0.565±0.044
Object-eff (cpu)	11.729±1.280	0.510±0.044	31.791±8.027	0.587±0.043	29.687±3.011	0.574±0.036

Table 22: Time cost of end-to-end inference on VQA. All cross-modal models are BERT-Base. On the SOTA number obtained with X152-C4 region features, the performance *keeps the same* when changing to the efficient way to extract the feature while the efficiency greatly improves on GPU. The efficient version does not lead to time saving on CPU, because nearly all inference time is taken by the backbone and C4 head and the time from NMS operations is nearly ignorable on CPU.

vision models dominate the inference time, especially for large models; (2) models based on grid-feature are faster than those based on region feature; (3) with our proposed fast inference trick, region-feature models are greatly sped up and their inference time can be brought to within 3 times of that of grid-feature models on GPU. We find that on CPU with a single thread, our class-agnostic trick does not lead to time saving, because nearly all inference time is taken by the backbone and C4 head and the time from NMS operations is nearly ignorable on CPU.