VirTex: Learning Visual Representations from Textual Annotations

Karan Desai and Justin Johnson

University of Michigan
{kdexd,justincj}@umich.edu

Abstract. The de-facto approach to many vision tasks is to start from pretrained visual representations, typically learned via supervised training on ImageNet. Recent methods have explored unsupervised pretraining to scale to vast quantities of unlabeled images. In contrast, we aim to learn high-quality visual representations from fewer images. To this end we revisit supervised pretraining, and seek data-efficient alternatives to classification-based pretraining. We propose VirTex – a pretraining approach using semantically dense captions to learn visual representations. We train convolutional networks from scratch on COCO Captions, and transfer them to downstream recognition tasks including image classification, object detection, and instance segmentation. On all tasks, VirTex yields features that match or exceed those learned on ImageNet – supervised or unsupervised – despite using up to ten times fewer images.

Keywords: Image captioning, pretraining, transfer learning

1 Introduction

Proposes training the CNN backbone with image captioning with fewer images than a simple image classifier on top of the backbone.

The prevailing paradigm for learning visual representations is first to *pretrain* a convolutional network [1, 2] to perform image classification on ImageNet [3, 4], then *transfer* the learned features to downstream tasks [5, 6]. This approach has been wildly successful, and has led to significant advances on a wide variety of computer vision problems such as object detection [7], semantic [8] and instance [9] segmentation, image captioning [10–12], and visual question answering [13, 14]. Despite its practical success, this approach is expensive to scale since the pretraining step relies on images annotated by human workers.

For this reason, there has been increasing interest in *unsupervised pretraining* methods that use unlabeled images to learn visual representations which are then transferred to downstream tasks [15–21]. Some recent approaches have begun to match or exceed supervised pretraining on ImageNet [22–26], and have been scaled to hundreds of millions [22, 24, 27, 28] or billions [25] of images.

Continuing to scale unsupervised pretraining to ever-larger sets of unlabeled images is an important scientific goal. But we may also ask whether there are alternate ways of pretraining that learn high-quality visual representations with *fewer* images. To do so, we revisit *supervised* pretraining and seek an alternative to traditional classification pretraining that uses each image more efficiently.

Unsupervised pretraining have shown impressive results on ImageNet. This paper revisits supervised training with broader targets than simple classifying with less

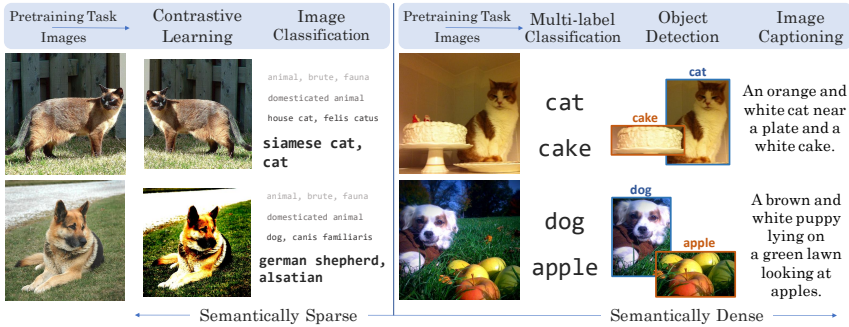

Fig. 1: **Comparison of pretraining tasks for learning visual representations:** Contrastive methods used for self-supervised learning [20, 23, 25, 26] provide a *semantically sparse* learning signal, encouraging different transforms of an image to have similar features. Classification associates each image with a single category, providing a learning signal of moderate semantic density. Multi-label classification and object detection increase semantic density by labeling multiple objects per image. Captions can mention many objects as well as attributes, relationships, and actions, giving a semantically dense learning signal.

Captions tell more about image and relation between the objects

We believe that using *language* as a supervisory signal is appealing due to its *semantic density*. Figure 1 compares different pretraining tasks for learning visual representations. Compared to unsupervised contrastive methods and supervised classification, captions provide a learning signal with more semantic content per image. Therefore we expect that textual annotations can be used to learn visual representations using fewer images than other approaches.

Another benefit of textual annotations is simplified data collection. To collect classification labels, typically human experts first build an ontology of categories [3, 4, 29, 30] then complex crowdsourcing pipelines are used to elicit labels from non-expert users [31, 32]. In contrast, natural language descriptions do not require an explicit ontology and can easily be written by non-expert workers, leading to a simplified data collection pipeline [33–35]. Large quantities of weakly aligned images and text can also be obtained from internet images [36–38].

In this paper we present an approach for learning **Vi**sual **r**epresentations from **Tex**tual annotations, abbreviated VirTex. Our approach is straightforward: first, we jointly train a convolutional network and a Transformer [39] *from scratch* to generate natural language captions for images. We then transfer the learned features to downstream visual recognition tasks.

Our main contribution is to show that natural language can provide supervision for learning transferable visual representations with better data-efficiency than other approaches. We train models from scratch on the COCO Captions dataset [35], and evaluate the learned features on downstream tasks including image classification, object detection, instance segmentation, and low-shot recognition. On all tasks, VirTex matches or exceeds the performance of existing methods for supervised or unsupervised pretraining on ImageNet, despite using up to 10× fewer images. Our code is available at `https://github.com/kdexd/virtex`

i.e. Pre training a CNN backbone

2 Related Work

Our work is related to recent efforts to move beyond supervised pretraining on ImageNet using alternate data sources or pretraining tasks.

Weakly Supervised Learning scales beyond supervised pretraining with a *quantity over quality* approach, and learns on large numbers of images with noisy labels taken from web services. YFCC-100M [40] comprises 10^8 images from Flickr annotated with user-provided tags. JFT-300M [41] uses Google's internal tooling to automatically label images from web signals, and has been used for large-scale weakly supervised learning [41, 42]. Weakly-supervised learning has also been studied on up to 3.5B Instagram images and tags [43]. These approaches learn visual representations with large quantities of images with low-quality labels; in contrast we focus on using fewer images with high-quality labels.

Semi-Supervised Learning learns visual representations from a mix of labeled and unlabeled images. S4L [44] uses self-supervised learning for unlabeled images, along with supervised learning for labeled images. Some recent approaches employ *teacher-student* learning [45, 46], where a teacher (trained on supervised data) predicts *pseudo-labels* on unlabeled data that the student mimics.

Self-Supervised Learning trains on *pretext tasks* for learning visual representations. Pretext tasks are defined on unlabeled images in the hopes that solving them requires some nontrivial semantic understanding; examples including solving jigsaw puzzles [47], predicting rotation [19], colorization [17, 18], inpainting [16], context prediction [15], clustering [27], and generative modeling [48].

More recent approaches employ the Noise Contrastive Estimation [49] learning paradigm, and use *contrastive losses* [50] which encourage similarity between image features under different random transformations on single input image [51]. Scaling these approaches can require memory banks [24, 52], queues [25], or TPU pods [26]. Other approaches use contrastive losses for context prediction [20, 23], mutual information maximization [21, 53, 54], predicting masked regions [55], and clustering [56]. Self-supervised learning uses unlabeled data, and has been scaled to millions [22, 24] or billions [25] of images. In contrast we aim to learn from *fewer* images via semantically dense textual annotations.

Large-Scale Language Models have become a dominant paradigm in natural language processing. Language models are first trained on large datasets of unlabeled text, and the learned representations are transferred to downstream tasks [57–59]. Since the pretraining step does not rely on human annotations, this approach has proven to be highly scalable, with more data and larger models generally leading to better performance on downstream tasks [60–63].

These advances show that language models trained on text learn transferable *textual* representations. Inspired by this success, we show that training image-conditional language models can also learn transferable *visual* representations.

Vision and Language Pretraining attempts to learn joint representations of paired visual and textual data that can be transferred to multimodal downstream tasks such as visual question answering [13, 14, 64, 65], visual reasoning [66, 67], referring expressions [68], and language-based image retrieval [34]. Inspired by

the success of BERT [59] in NLP, several recent methods use Transformers [39] to learn transferable joint representations of images and text [69–75].

These methods employ complex pretraining pipelines: they typically (1) start from a CNN pretrained for ImageNet classification; (2) extract region features using an object detector fine-tuned on Visual Genome [76], following [77]; (3) optionally start from a pretrained language model, such as BERT [59]; (4) combine the models from (2) and (3), and train a multimodal transformer on Conceptual Captions [36]; (5) fine-tune the model from (4) on the downstream task. In this pipeline, all vision and language tasks are downstream from the initial visual representations learned on ImageNet. In contrast, we train models from scratch for image captioning, and transfer the learned visual features to downstream visual recognition tasks. We believe that we are the first to demonstrate that vision and language pretraining can be used to learn competitive visual features.

3 Method

Given a dataset of images with captions, our goal is to learn visual representations of images that can be transferred to downstream visual recognition tasks. As shown in Figure 1, captions can compactly express different kinds of semantic image content, including the presence of objects (*cat*, *plate*, *cake*); attributes of objects (*orange and white cat*; spatial arrangement of objects (*cat near a plate*); and actions (*looking at apples*). Learned visual representations that capture such rich semantic content should be useful for many downstream vision tasks.

To this end, we train *image captioning* models to predict captions from images. As shown in Figure 2, our model has two components: a *visual backbone* and a *textual head*. The visual backbone extracts visual features from images. The textual head accepts visual features and predicts a caption $C = (c_1, \ldots, c_T)$ token by token. The textual head performs bidirectional captioning (*bicaptioning*): it comprises a *forward model* that predicts tokens left-to-right, and a *backward model* that predicts right-to-left. All model components are randomly initialized, and jointly trained to maximize the log-likelihood of the correct caption tokens

$$\mathcal{L}(\theta, \phi) = \sum_{t=1}^{T} \log(c_t \mid c_1, \ldots, c_{t-1}; \phi_f, \theta) + \sum_{t=1}^{T} \log(c_t \mid c_{t+1}, \ldots, c_T; \phi_b, \theta) \quad (1)$$

where θ, ϕ_f, and ϕ_b are the parameters of the visual backbone, forward, and backward models respectively. After training, the textual head is discarded and the visual backbone is transferred to downstream visual recognition tasks.

Language Modeling: Our system is inspired by recent work in NLP using language models to learn transferable text representations [57–60, 62, 63]. Of these, some use unidirectional [58, 60] or bidirectional [57] language models that predict tokens one by one. However, following BERT [59], many large-scale models [62, 63] instead use *masked language models* (MLMs): some tokens are randomly masked and are predicted by the model. We performed preliminary experiments with MLMs, but like [59] we observed that MLMs converge

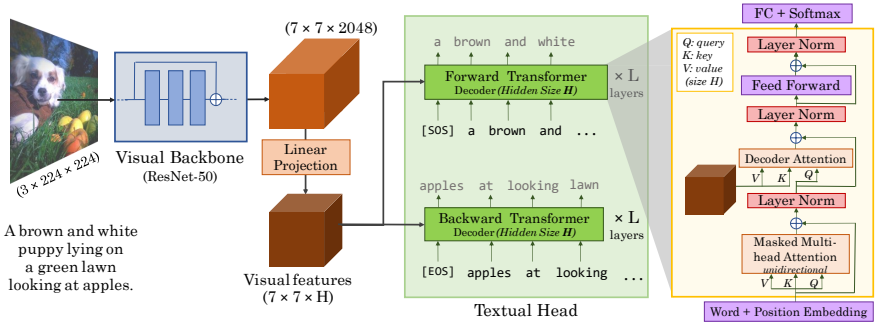

Fig. 2: **VirTex pretraining setup:** Our model consists of a *visual backbone* (ResNet-50), and a *textual head* (two unidirectional Transformers). The visual backbone extracts image features, and textual head predicts captions via bidirectional language modeling (*bicaptioning*). The Transformers perform masked multiheaded self-attention over caption features, and multiheaded attention over image features. Our model is trained end-to-end from scratch. After pretraining, the visual backbone is transferred to downstream visual recognition tasks.

more slowly than directional models. During each training iteration, MLMs only predict a subset of each caption's tokens, while directional models predict all tokens. We hypothesize that this gives a weaker training signal, thus slower convergence. Due to computational constraints, we focus on directional models and leave MLMs to future work. Emperically we found that bidirectional language models learn slightly better visual representations than unidirectional models.

Visual Backbone: The visual backbone is a convolutional network which computes visual features of images. It inputs raw image pixels, and outputs a spatial grid of image features. During pretraining, these features are used to predict captions. In downstream tasks, we either train linear models on features extracted from the visual backbone, or fine-tune the visual backbone end-to-end.

In principle we could use any convolutional network architecture for the visual backbone. In our experiments we use a standard ResNet-50 [2] as the visual backbone to facilitate comparison with prior work on supervised and unsupervised representation learning [24, 25]. It accepts a 224×224 image and produces a 7×7 grid of 2048-dimensional features after the final convolutional layer. During pretraining, we apply a linear projection layer to the visual features before passing them to the textual head to facilitate decoder attention over visual features. This projection layer is not used in downstream tasks.

Textual Head: The textual head receives features from the visual backbone and predicts captions for images. It provides a learning signal to the visual backbone during pretraining. Our overall goal is not to predict high-quality captions, but instead to learn transferable visual features.

The textual head comprises two identical language models which predict captions in forward and backward directions respectively. Following recent advances in language modeling [59], we use Transformers [39], which use multiheaded self-attention both to propagate information along the sequence of caption tokens,

as well as to fuse visual and textual features. We closely follow the decoder from [39], but use GELU [78] rather than ReLU, following [58, 59]. We briefly review the architecture here; refer to [39] for a more complete description.

During training, the forward model receives two inputs: image features from the visual backbone, and a caption describing the image. Image features are a matrix of shape $N_I \times D_I$ giving a D_I-dimensional vector for each of the $N_I = 7 \times 7$ positions in the final layer of the visual backbone. The caption $C = (c_1, \ldots, c_T)$ is a sequence of T tokens, where $c_1 =$ `[SOS]` is a start-of-sequence token. The forward model is trained to predict a shifted sequence $C' = (c'_1, \ldots, c'_T) = (c_2, \ldots, c_{T+1})$ where $c_{T+1} =$ `[EOS]` is an end-of-sequence token. The prediction for $c'_t = c_{t+1}$ is *causal*, depending only on $c_1, \ldots, c_t$ and the visual features. The backward model is similar, but predicts right-to-left instead.

Tokens of C are first converted to vectors using learned token and positional embeddings, followed by elementwise sum, layer normalization [79] and dropout [80]. These per-token vectors are then processed by a sequence of Transformer layers. As shown in Figure 2, each layer performs masked multiheaded self-attention over token vectors, multiheaded attention between token vectors and image vectors, and applies a fully-connected network to each token vector. These three operations are each followed by layer normalization and dropout and wrapped in a residual connection. Per-token vectors interact only through self-attention; the masking in this operation maintains the causal structure of the final predictions. After the final Transformer layer, a linear layer is applied to each vector to predict unnormalized log-probabilities over the token vocabulary.

The forward and backward models consist of independent Transformer layers. However they share the same token embedding matrix (similar to [57]) which is also reused at the output layers of each model (similar to [81, 82]).

Model Size: Several architectural hyperparameters control the size of our textual head. We can control the *width* of each Transformer layer by varying its *hidden dimension* H, the number of *attention heads* A used in multiheaded attention, and the *feedforward dimension* F of the elementwise fully-connected network. We follow [59] and always set $A = H/64$ and $F = 4H$; this allows us to control the width of our textual head by varying H. We can also control the *depth* of our textual head by varying the number of transformer layers L.

Tokenization: We tokenize captions with SentencePiece [83] using the BPE algorithm [84]. Prior to tokenization we lowercase and strip accents from captions. We build a vocabulary of 10K tokens, including boundary (`[SOS]`, `[EOS]`) and out-of-vocab (`[UNK]`) tokens. Following [58] we restrict subword merges between letters and punctuation to prevent redundant tokens such as `dog?` and `dog!`. Compared to basic tokenization schemes often used for image captioning that split on whitespace [10, 11], BPE makes fewer linguistic assumptions, exploits subword information, and results in fewer out-of-vocab tokens.

Training Details: We train on the `train2017` split of the COCO Captions dataset [35], which provides $118K$ images with five captions each. During training we apply standard data augmentation: we randomly crop to 20-100% of the original image size, apply color jitter (brightness, contrast, saturation, hue), and

normalize using the ImageNet mean color. We also apply random horizontal flips, also interchanging the words 'left' and 'right' in the caption.

We train using SGD with momentum 0.9 [85, 86] and weight decay 10^{-4} wrapped in LookAhead [87] with $\alpha = 0.5$ and 5 steps. Following [59], we do not apply weight decay to layer normalization and bias parameters in Transformers. We perform distributed training across 8 GPUs with batch normalization [88] per GPU, following [22]. We train with a batch size of 256 images (32 per GPU) for 500K iterations (≈1080 epochs). We use linear learning rate warmup [22] for the first 10K iterations followed by cosine decay [89] to zero. We found that the visual backbone required a higher LR than textual head for faster convergence. The visual backbone uses a max learning rate of 2×10^{-1}; the textual head uses 10^{-3}. We use PyTorch [90] and NVIDIA Apex for mixed-precision training [91].

We observe that performance on image captioning has positive but imprecise correlation with performance on downstream visual recognition tasks. We thus perform *early stopping* based on the performance of our visual backbone on downstream PASCAL VOC [92] linear classification (see Section 4.1) since it is fast to evaluate and correlates well with our other downstream tasks.

4 Experiments

In our experiments we aim to show that the semantic density of language annotations can be used to learn high-quality visual representations using fewer images than other approaches. First, as described in Section 3, we randomly initialize and jointly train a ResNet-50 [2] visual backbone and a Transformer [39] textual head to perform bicaptioning on the COCO Captions [35] dataset. We then adapt the visual backbone for downstream recognition tasks.

Following prior works [22, 24, 25], we select downstream tasks based on two common mechanisms for transfer learning: where the visual backbone is either used as (a) frozen feature extractor, or (b) weight initialization for fine-tuning.

Baselines: We compare the performance of VirTex-pretrained visual backbone with other methods for learning visual features.

- **Random**: Uses no pretrained visual features; the visual backbone is randomly initialized and trained on the downstream task.
- **ImageNet-supervised (IN-sup)**: The visual backbone is trained for image classification on the ILSVRC 2012 [4] `train` split (1.28M images).[1]
- **PIRL** [24]: A self-supervised method for learning visual features by encouraging learned representations to invariant to transformations.[2]
- **MoCo** [25]: A self-supervised contrastive method for learning visual features that scales to large batches using a momentum-based encoder and a queue. We consider two different MoCo baselines:
  - **MoCo-IN**: trained on ImageNet – publicly available pretrained model.
  - **MoCo-COCO**: trained on COCO with default hyperparameters.[3]

[1] We use pretrained model from `torchvision` package: `github.com/pytorch/vision`
[2] We adopt numbers from [24] as the code and models are not yet made public.
[3] Pretrained model and code used from: `github.com/facebookresearch/moco`

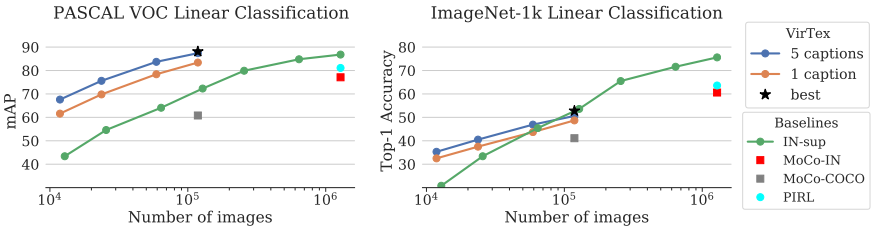

Fig. 3: **Linear classification results:** We learn visual features using varying amounts of captioning data from COCO, and varying amounts of classification data from ImageNet. We evaluate these features by training linear classifiers. Learning features via language is much more data-efficient than other methods.

For fair comparison, all methods use the same ResNet-50 architecture. Apart from model architecture, there are certain differences across baselines and our setup. These stem from difference in training setup, such as different epochs, LR schedules and weight decay regularization. We try to ensure minimal difference in these factors, and report any such differences where applicable.

4.1 Image Classification with Linear Models

We measure the quality of learned features from VirTex pretraining by training linear models on features extracted from frozen visual backbones. We evaluate on two image classification datasets: PASCAL VOC [92] and ImageNet-1k [3, 4], following evaluation protocols consistent with our chosen baselines.

PASCAL VOC: We follow the same protocol as [22, 24]: we train on VOC07 `trainval` split ($\sim 9K$ images, 20 classes) and report mAP on `test` split. We extract a 7×7 grid of 2048-dimensional features from the last layer of visual backbone and downsample them using adaptive average pooling to a 2×2 grid. We flatten and L2-normalize these to yield 8192-dimensional features. We train per-class SVMs, for cost values $C \in \{0.01, 0.1, 1, 10\}$. We choose best C per class by 3-fold cross-validation. All other SVM hyperparameters are same as [22].

ImageNet-1k: We follow the same protocol as [25]: we train on the ILSVRC 2012 `train` split and report top-1 center crop accuracy on `val` split. We train a linear classifier (fully connected layer + softmax) on 2048-dimensional global average pooled features extracted from the last layer of visual backbone. We train with batch size 256 distributed across 8 GPUs for 100 epochs. We use SGD with momentum 0.9, weight decay 0 and learning rate 0.3[4], which is divided by 10 at epochs 60 and 80. Refer Appendix A.1 for more details.

Data Efficiency: As shown in Figure 1, we believe that the semantic density of captions should allow our method to learn effective visual features from fewer training images than other methods. To test our hypothesis, we compare the linear classification performance of VirTex-pretrained backbones and ImageNet-supervised models trained using varying amount of instances from

[4] Following [25], we conduct a small sweep to find the optimal learning rate. We set learning rate as $x \times 10^y$, where $x \in \{1, 2, 3\}$ and $y \in \{-2, -1, 0, 1\}$.

COCO Captions and ImageNet-1k respectively. Specifically, we train 4 VirTex models using $\{10, 20, 50, 100\}\%$ of COCO Captions (118K images) and 7 ResNet-50 models using $\{1, 2, 5, 10, 20, 50, 100\}\%$ of ImageNet-1k (1.28M images). Note that COCO Captions provides 5 captions per image, which effectively increases image-caption pairs by five-fold. Hence for fair comparison, we also train 4 additional VirTex models using only one randomly selected caption per image.

All VirTex models use same textual heads with $L = 1$ and $H = 1024$. Refer Appendix A.2 for more details on data sampling and training hyperparameters.

Results: We show results in Figure 3. On VOC07, VirTex-pretrained backbone trained using entire COCO Captions outperforms ImageNet-supervised model trained using entire ImageNet (mAP **87.4** vs **86.8**) despite using $10\times$ fewer images (118K vs 1.28M). When using similar amount of images, VirTex models consistently outperform ImageNet-supervised models (**blue**, **orange** vs **green**). These trends indicate that learning via textual annotations is more data-efficient than classification labels. We also observe that given a constant number of captions for training, it is better to spread them over more images: training on 50% of COCO images with one caption per image significantly outperforms using 10% of images with five captions per image (mAP **78.1** vs **67.6**).

Linear classification on ImageNet is an extremely unfair comparison for VirTex-pretrained backbones vs. ImageNet-supervised models, since the latter is trained explicitly for the downstream task. Even so, when using fewer than 100K images, VirTex models consistently outperform ImageNet-supervised models using equivalent number of labeled images (**blue** vs **green**), similar to VOC07. Our best model trained on all COCO captions closely matches ImageNet-supervised model trained on 10% ImageNet (**52.8** vs. **53.6**, 118K vs 128K images).

4.2 Ablations

The preceeding linear classification experiments demonstrate the effectiveness and data-efficiency of our approach to learning visual features from language. In this section we perform ablation studies to isolate the effects of our pretraining setup and modeling decisions, and uncover performance trends to seed intuition for future work. In these experiments, we evaluate learned features on ImageNet-1k and PASCAL VOC as described in section 4.1.

How do other pretraining tasks compare with bicaptioning? We use captions due to their semantic density, and bicaptioning as it gives a strong training signal per instance. Here we test this intuition by comparing our bicaptioning model with features learned from four other tasks on COCO:

- **Forward Captioning** performs only unidirectional left-to-right language modeling by removing the backward model from the textual head.
- **Token Classification** ignores the linguistic structure of captions, and treats them as an unordered set of tokens. We replace the textual head with a linear layer that predicts a classification score for each vocabulary token. For a caption with K unique tokens, we train the model model to predict a K-hot vector with values $1/K$ with a KL-Divergence loss, similar to [43].

Pretraining Task	VOC07	IN-1k
Bicaptioning	**88.1**	**52.8**
Forward Captioning	87.7	50.7
Token Classification	**88.1**	46.4

(a) Pretraining tasks using captions.

Pretraining Task	VOC07	IN-1k
Bicaptioning	**88.1**	**52.8**
Multi-Label Classif.	85.3	46.4
Instance Segmentation	81.0	50.1

(b) Pretraining tasks using boxes/masks.

Table 1: **Ablations: Pretraining Tasks.** We compare bicaptioning with four other tasks on COCO. **Table (a):** bicaptioning outperforms forward captioning, and naive token classification (which does not model the language). **Table (b):** bicaptioning outperforms other pretraining tasks which use box/mask annotations, indicating that using captions results in better quality of visual features.

- **Multi-Label Classification** uses COCO object detection annotations instead of captions. The setup is identical to Token Classification, but each image's label set is the set of object classes that appear in the image.
- **Instance Segmentation** trains Mask R-CNN [9] with a ResNet-50-FPN backbone [93] from scratch on COCO, following [94]. We use the trained model provided by the Detectron2 model zoo [95], and extract the backbone for use in downstream tasks.

For captioning and bicaptioning, we use textual heads with $L = 1, H = 2048$. Other than instance segmentation, we train models for all tasks with the same optimization setup as our model (see Section 3). Unlike the comparisons in Section 4.1, all of these models are trained on the same set of images which allows us to more precisely pinpoint the effect of different training objectives.

Results are shown in Table 1. Bicaptioning outperforms forward captioning; we hypothesize that bidirectional modeling gives a denser supervisory signal from each caption, improving the visual features. Bicaptioning and forward captioning both outperform Token Classification, demonstrating that learning to model the sequential structure of language also results in better visual features. All three models trained with language outperform the two models trained with object labels, showing the benefit of learning from the semantically denser language.

Do bigger transformers help? Prior work in language modeling has shown that larger Transformers tend to learn better *textual* features [60–63]. We investigate whether the same holds for our VirTex models: does a larger transformer in the textual head cause the visual backbone to learn better *visual* features?

As discussed in Section 3, we may scale our textual head by increasing its *width* (hidden size H) or its *depth* (number of layers L). We investigate both:

- **Scaling width:** We train models with fixed depth $L = 1$ and increasing widths $H \in \{512, 768, 1024, 2048\}$; results are shown in Table 2 (left). Increasing width consistently improves downstream classification performance.
- **Scaling depth:** We train models with fixed width $H = 1024$ and increasing depths $L \in \{1, 2, 3, 4\}$; results shown in Table 2 (right). Similar to width, increasing depth consistently improves downstream classification performance.

We could not fit larger transformers beyond Table 2 on our GPUs. However, based on these empirical observations, we think that even larger widths and depths may continue to improve performance. We hope that huge transformers,

Depth L	Width H	VOC07	IN-1k
1	512	86.6	49.1
1	768	86.8	50.0
1	1024	87.4	50.6
1	2048	**88.1**	**52.8**

Depth L	Width H	VOC07	IN-1k
1	1024	87.4	50.6
2	1024	87.4	50.9
3	1024	87.5	51.2
4	1024	**87.7**	**52.1**

Table 2: **Ablations: Transformer Size.** Comparison across varying Transformer width (hidden size H) and depth (layers L). All models use ResNet-50 as the visual backbone. We show that larger transformers improve downstream linear classification performance – both, wider (**left**) and deeper (**right**).

Visual Backbone	VOC07	IN-1k
ResNet-50	87.4	50.6
ResNet-50 w2×	**87.5**	**51.0**
ResNet-50 w2× (IN)	87.6	76.7

Visual Backbone	VOC07	IN-1k
ResNet-50	87.4	50.6
ResNet-101	**87.7**	**51.7**
ResNet-101 (IN)	87.3	77.1

Table 3: **Ablations: Backbone Size.** We show that bigger visual backbones improve downstream linear classification performance – both, wider (**left**) and deeper (**right**). All VirTex models use textual heads with depth $L = 1$ and width $H = 1024$. VirTex continues to closely match, or outperform IN-sup models (IN: gray) on VOC07 with bigger backbones, similar to ResNet-50 (Figure 3).

comparable to the size of BERT [59] and GPT-2 [60] will help when scaling up to orders of magnitude larger, more noisy image-text paired datasets [36–38].

Do bigger visual backbones help? Bigger visual backbones tend to give improvements on many visual recognition tasks [2, 9, 96]. We investigate whether scaling backbone capacity in VirTex models can improve performance on downstream tasks. We train two VirTex models with bigger visual backbones and compare them with base model using ResNet-50 – (a) ResNet-50 w2× [97] (2× channel width), and (b) ResNet-101 (2× depth). Both variants use textual head with $L = 1, H = 1024$. Results are shown in Table 3. Using higher-capacity backbones improve downstream performance on both PASCAL VOC and ImageNet-1k. We also include ImageNet-supervised models[5] for reference (gray) – we observe that VirTex models with bigger backbones closely match, or outperform ImageNet-supervised models on PASCAL VOC, as with ResNet-50 (Figure 3).

4.3 Fine-tuning Tasks for Transfer

So far we have evaluated VirTex using features extracted from *frozen* visual backbones. Another common mechanisms for transfer learning is *fine-tuning*, where the entire visual backbone is updated for the downstream task.

We evaluate features learned using VirTex on four downstream tasks with fine-tuning: (a) Instance Segmentation on COCO [29]; (b) Instance Segmentation on LVIS [30]; and (c) Object Detection on PASCAL VOC [92]; (d) Fine-grained Classification on iNaturalist 2018 [98]. In all these fine-tuning experiments, we

[5] Similar to ResNet-50, we use pretrained models provided with `torchvision`.

Method	COCO Box AP			COCO Mask AP			LVIS Mask AP		
	AP_{all}	AP_{50}	AP_{75}	AP_{all}	AP_{50}	AP_{75}	AP_{all}	AP_{50}	AP_{75}
Random Init	36.7	56.7	40.0	33.7	53.8	35.9	22.5	34.8	23.8
IN-sup	41.1	62.0	44.9	37.2	59.1	40.0	24.5	38.0	26.1
MoCo-IN [25]	40.8▼	61.6▼	44.7	36.9▼	58.4▼	39.7▼	24.1▼	37.4▼	25.5▼
MoCo-COCO	38.5▼	58.5▼	42.0▼	35.0▼	55.6▼	37.5▼	23.1▼	35.3▼	24.9▼
VirTex (ours)	40.9	61.7▼	44.8	36.9▼	58.4▼	39.7▼	25.4▲	39.0▲	26.9▲

Table 4: **Instance Segmentation on COCO and LVIS:** We use Mask R-CNN with ResNet-50-FPN backbones. **(a) Random, MoCo-COCO:** On both COCO and LVIS, VirTex performs best among methods which only use COCO (118K images) during pretraining and fine-tuning. **(b) IN-supervised, MoCo-IN:** On COCO, VirTex closely matches methods which also use ImageNet (1.28M images) during pretraining, while outperforming them on LVIS.
▼,▲: Performance difference is ±**0.3** or larger with respect to IN-supervised.

use the VirTex model with ResNet-50 visual backbone, and textual head with depth $L = 1$, width $H = 2048$ (Table 2 left, row 4). We compare with our chosen baselines for learning visual features, as described at the start of Section 4.

We follow the same evaluation protocol as MoCo [25] for all four tasks. For tasks (a, b, c), our ImageNet-supervised results are slightly better than those reported in [25] – we use pretrained ResNet-50 model from `torchvision`, whereas they used the MSRA ResNet-50 model from Detectron [99]. We use Detectron2 [95] for these tasks, and describe implementation details which differ from its default settings. Refer Appendix A.3 for full details.

Instance Segmentation on COCO: We train Mask R-CNN [9] models with ResNet-50-FPN backbones [93]. We initialize backbone with pretrained (or random) weights, train on `train2017` split, and evaluate on `val2017` split. During fine-tuning we update all backbone layers, including BN layers which are synchronized across GPUs [100] (*SyncBN*). We also use SyncBN in FPN layers. We train with batch size 16 distributed across 8 GPUs, following 2× schedule (180K iterations with initial LR 0.02, multiplied by 0.1 at iterations 120K and 160K).

Results are shown in Table 4. VirTex *significantly outperforms* methods which only use COCO (118K images) during pretraining and fine-tuning – Random Init and MoCo-COCO. VirTex matches MoCo-IN, but is slightly outperformed by IN-sup. However, VirTex is significantly data-efficient – these methods also use ImageNet (1.28M images) during pretraining, unlike VirTex.

Instance Segmentation on LVIS: The LVIS dataset provides instance segmentation labels for a long tail of 1230 entry-level object categories, and stresses the ability to recognize many object types from few training samples.

Similar to COCO, we train Mask R-CNN models with ResNet-50-FPN backbones. We train on `train_v0.5` and evaluate on `val_v0.5` split. Like [25], we keep BN frozen for IN-sup baseline; other models are fine-tuned with SyncBN. We train with 2× schedule as COCO, use class resampling and test-time hyperparameters (0.0 score threshold and 300 detections per image) same as [30].

Method	AP_{50}	AP_{all}	AP_{75}
Random Init	60.2	33.8	33.1
IN-sup	81.6	54.3	59.7
MoCo-IN [25]	81.5	55.9▲	62.6▲
PIRL♣ [24]	80.7▼	54.0	59.7
MoCo-COCO	75.4▼	47.5▼	51.1▼
VirTex (ours)	81.4	55.6▲	61.5▲

(a) PASCAL VOC object detection

Method	Top-1 acc.
Random Init	61.4
IN-sup	64.9
MoCo-IN [25]	62.9▼
IN-sup (10% IN)♠	59.7▼
MoCo-COCO	61.1▼
VirTex (ours)	63.6▼

(b) iNaturalist 2018 classification

Table 5: **(a) Object Detection on PASCAL VOC:** We train Faster R-CNN detectors with ResNet-50-C4 backbones. All results are the average of five trials. VirTex closely matches MoCo-IN and outperforms all other methods which use either COCO or ImageNet (10× larger than COCO) during pretraining.
(b) Fine-grained Classification on iNaturalist 2018: VirTex performs much better than Random Init and outperforms MoCo-COCO, but falls behind methods which use ImageNet during pretraining.
♣: Uses longer training schedule and keeps BN frozen during fine-tuning.
♠: Trained with 10% ImageNet, refer Section 4.1.

Results are shown in Table 4. Despite being trained only on COCO images, VirTex *significantly outperforms* all baselines – including those which also use ImageNet during pretraining (IN-sup and MoCo-IN).

Object Detection on PASCAL VOC: We train Faster R-CNN [101] models with ResNet-50-C4 backbones. We train on `trainval07+12` split, and evaluate on `test2007` split. Like COCO, we fine-tune all models with SyncBN. We train for 24K iterations, including linear LR warmup for first 100 iterations. We set the maximum LR as 0.02, which is multipled by 0.1 at iterations 18K and 22K. We distribute training across 8 GPUs, with batch size 2 per GPU. C4 backbones have high memory footprint, which exceeds our 12 GB GPU capacity. We compensate this by reducing memory usage through gradient checkpointing [102, 103].

Results are shown in Table 5 (a). Since PASCAL VOC is a small dataset, models trained from scratch (Random Init) perform much worse than those initialized with pretrained backbones. Among methods which use COCO for pretraining, VirTex significantly outperforms MoCo-COCO. VirTex also outperforms methods which use ImageNet (10× larger than COCO) during pretraining – IN-sup and PIRL, but is slightly outperformed by MoCo-IN.

Fine-grained Classification on iNaturalist 2018: The iNaturalist 2018 dataset provides labeled images for 8142 fine-grained categories, with a long-tailed distribution. We fine-tune the pretrained ResNet-50 with a linear layer end-to-end. We train on `train2018` split and evaluate on `val2018` split. We follow ResNet-50 training setup from `torchvision` – we train for 100 epochs using SGD with momentum 0.9 and weight decay 10^{-4}, and batch size 256 distributed across 8 GPUs. Fine-tuning uses LR 0.025 (and Random Init uses 0.1), which is multiplied by 0.1 at epochs 70 and 90. Refer Appendix A.4 for more details.

Results are shown in Table 5 (b). VirTex outperforms Random Init, MoCo-COCO and MoCo-IN, but is outperformed by IN-sup. To control difference in

Backbone	Depth	Width	CIDEr	SPICE	Iter
ResNet-50	1	512	89.3	17.8	496K
ResNet-50	1	768	90.1	18.0	500K
ResNet-50	1	1024	89.5	**18.1**	**480K**
ResNet-50	1	2048	**90.3**	**18.1**	500K
ResNet-50	1	1024	89.5	**18.1**	480K
ResNet-50	2	1024	90.5	17.5	**420K**
ResNet-50	3	1024	**92.0**	17.8	472K
ResNet-50	4	1024	91.9	17.6	**464K**
ResNet-50 w2×	1	1024	88.2	17.6	**378K**
ResNet-101	1	1024	**94.0**	**18.5**	498K

Fig. 4: **Image Captioning Results. Left:** We evaluate VirTex models on COCO `val2017` split. We report **CIDEr** [104] and **SPICE** [105] metrics for the checkpoint chosen by best VOC07 mAP (at iteration **Iter**). Changes that improve captioning also tend to improve downstream performance – except few models which are chosen at earlier iterations (**orange**), indicating that the textual head is slightly under-trained. **Right:** Qualitative results of captions predicted by our best model. it describes objects in coherent language. The second column shows failure modes: word repetitions and miscounting objects.

dataset size, we compare VirTex with IN-sup trained with 10% ImageNet (118K images vs. 128K images). VirTex outperforms this model (which itself is slightly worse than Random Init), yet again demonstrating its data-efficiency. We do not compare with PIRL; it keeps the backbone frozen, hence not directly comparable.

4.4 Image Captioning

Our goal is to learn transferable visual features via textual supervision. To do so, we use image captioning as a pretraining task. Although our goal is not to improve the state-of-the-art in image captioning, in Figure 4 we show quantitative and qualitative results of our image captioning models trained from scratch on COCO. The quantitative results show the same general trends as Section 4.2: both, bigger transformers and bigger visual backbones generally perform better, except a few models (**orange**) which were *early-stopped* (based on VOC07 performance) much earlier than others – indicating that the textual head is slightly under-trained. This shows that improvements in our pretraining task tend to correlate with improvements in downstream visual recognition tasks.

5 Conclusion

We have shown that learning visual representations using textual annotations can be a competitive alternate to methods based on supervised classification and self-supervised learning on ImageNet. Our downstream tasks focus solely on the vision backbone – future works can explore downstream tasks which may transfer both the visual backbone and the textual head. Finally, the usage of captions opens a clear pathway to scaling our approach to web-scale image-text pairs, which are orders of magnitude larger, albeit more noisy than COCO.

Acknowledgments

We thank Harsh Agrawal, Mohamed El Banani, Richard Higgins, Nilesh Kulkarni and Chris Rockwell for helpful discussions and feedback on the draft. We thank Ishan Misra for discussions regarding PIRL evaluation protocol; Saining Xie for discussions about replicating iNaturalist evaluation as MoCo; Ross Girshick and Yuxin Wu for help with Detectron2 model zoo; Georgia Gkioxari for suggesting the Instance Segmentation pretraining task ablation; and Stefan Lee for suggestions on figure aesthetics. We thank Jia Deng for access to extra GPUs during project development; and UMich ARC-TS team for support with GPU cluster management. Finally, we thank all the Starbucks outlets in Ann Arbor for many hours of free WiFi. This work was partially supported by the Toyota Research Institute (TRI). However, note that this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

Appendix

A Additional Implementation Details

As mentioned in Sections 4.1 and 4.3, our evaluation protocol is consistent with our chosen baselines [24, 25]. Here, we describe additional details for all downstream tasks to aid an exact replication of our setup. Most of these details are same as prior works, we report differences where applicable.

A.1 Image Classification with Linear Models

PASCAL VOC: For feature extraction, each image in `trainval` and `test` split is resized to 224×224, followed by normalization by ImageNet color mean. Prior works train per-class SVMs for $C \in [2^{-19}, 2^{-4}] \cup [10^{-7}, 10^{-2}]$ (26 values), and choose best SVM based on 3-fold cross-validation. In our initial evaluations, we observed that the best performing SVMs are typically trained with cost $C \in \{0.01, 0.1, 1.0, 10.0\}$. Based on this observation, we only train SVMs on these four cost values. We use the same hyperparameters as [22]: we use scikit-learn [106] package with LIBLINEAR [107] backend, default parameters are: `LinearSVC(penalty='l2', class_weight={1: 2, -1: 1}, max_iter=2000, loss='squared_hinge', tol=1e-4, dual=True)`.

ImageNet-1k: During training, we perform simple data augmentation: random resized crop of 20-100% of the original image with random aspect ratio between 4:3 and 3:4, followed by resizing to 224×224, random horizontal flip and normalization by ImageNet color mean. During evaluation, we resize the shortest edge to 256 and take a center crop of 224×224. The weights of fully connected layer are randomly initialized as $N(0.0, 0.01)$, and biases are initialized as 0.

Note that we perform a small LR sweep separately for our best VirTex model, MoCo-COCO and IN-sup models. For Figure 3, best LR values for VirTex models is 0.3 (as mentioned in Section 4.1), MoCo-COCO is 30.0 (similar to MoCo-IN from [25]) and IN-sup is 0.1.

A.2 Data Efficiency

Data Sampling: We train VirTex models using 10%, 20%, 50%, and 100% of COCO Captions [35] dataset. We subsample training instances randomly – we do not use bounding box annotations to enforce uniform class distribution.

We do the same for our ImageNet-pretrained baselines – we randomly sample 1%, 2% 5%, 10%, 20%, 50%, and 100% of ImageNet. Here, we sample in a way which keeps class distribution nearly same; *this may put ImageNet-supervised models at a minor advantage*. Note that ImageNet is 10× larger than COCO (1.28M vs. 118K images) – in terms of number of images, 1% IN $\simeq$ 10% COCO; 5% IN $\simeq$ 50% COCO; and 10% IN $\simeq$ 100% COCO.

Training Details for ImageNet-supervised models: We train our ImageNet-supervised models by following the *exact* hyperparameters used to train the publicly available ResNet-50 in `torchvision` (trained on 100% ImageNet). We use SGD with momentum 0.9 and weight decay 10^{-4}. We use a batch size of 256, and perform distributed training across 8 GPUs (batch size 32 per GPU). We train for 90 epochs, with an initial learning rate of 0.1, multiplied by 0.1 at epochs 30 and 60. Note that we keep the number of epochs same when training with different dataset sizes; which results in models trained with less instances undergo lesser iterations (otherwise they tend to overfit). Likewise for VirTex models, we scale our training iterations as per the amount of training instances.

A.3 Fine-tuning tasks for Transfer: Detectron2

We described main details for downstream fine-tuning tasks in Section 4.3. Here, we provide config files in Detectron2 [95] format to fully replicate the setup of our downstream fine-tuning tasks on COCO, LVIS and PASCAL VOC. We apply modifications on top of base config files available at:
github.com/facebookresearch/detectron2 @ 54d8e7

Instance Segmentation on COCO:

```
_BASE_: "Base-RCNN-FPN.yaml"
INPUT:
  FORMAT: "RGB"
DATASETS:
  TRAIN: ("coco_2017_train",)
  TEST: ("coco_2017_val",)
MODEL:
  WEIGHTS: "Loaded externally"
  MASK_ON: True
  PIXEL_MEAN: [123.675, 116.280, 103.530]
  PIXEL_STD: [58.395, 57.120, 57.375]
  BACKBONE:
    FREEZE_AT: 0
  RESNETS:
    DEPTH: 50
    NORM: "SyncBN"
    STRIDE_IN_1X1: False
  FPN:
    NORM: "SyncBN"
SOLVER:
  IMS_PER_BATCH: 16
  BASE_LR: 0.02
  STEPS: (120000, 160000)
```

```
  MAX_ITER: 180000
TEST:
  PRECISE_BN:
    ENABLED: True
```

Instance Segmentation on LVIS:

```
_BASE_: "Base-RCNN-FPN.yaml"
INPUT:
  FORMAT: "RGB"
DATASETS:
  TRAIN: ("lvis_v0.5_train",)
  TEST: ("lvis_v0.5_val",)
DATALOADER:
  SAMPLER_TRAIN: "RepeatFactorTrainingSampler"
  REPEAT_THRESHOLD: 0.001
MODEL:
  WEIGHTS: "Loaded externally"
  MASK_ON: True
  PIXEL_MEAN: [123.675, 116.280, 103.530]
  PIXEL_STD: [58.395, 57.120, 57.375]
  BACKBONE:
    FREEZE_AT: 0
  RESNETS:
    DEPTH: 50
    NORM: "SyncBN"
    STRIDE_IN_1X1: False
  FPN:
    NORM: "SyncBN"
  ROI_HEADS:
    NUM_CLASSES: 1230
    SCORE_THRESH_TEST: 0.0
SOLVER:
  IMS_PER_BATCH: 16
  BASE_LR: 0.02
  STEPS: (120000, 160000)
  MAX_ITER: 180000
TEST:
  DETECTIONS_PER_IMAGE: 300
  PRECISE_BN:
    ENABLED: True
```

For IN-sup baseline, change `RESNETS.NORM` to `"FrozenBN"` and `FPN.NORM` to `""`.

Object Detection on PASCAL VOC:

```
_BASE_: "Base-RCNN-C4.yaml"
INPUT:
  FORMAT: "RGB"
  MIN_SIZE_TRAIN: (480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800)
DATASETS:
  TRAIN: ("voc_2007_trainval", "voc_2012_trainval")
  TEST: ("voc_2007_test",)
MODEL:
  MASK_ON: False
  WEIGHTS: "Loaded externally"
  PIXEL_MEAN: [123.675, 116.280, 103.530]
  PIXEL_STD: [58.395, 57.120, 57.375]
  BACKBONE:
    FREEZE_AT: 0
  RESNETS:
    DEPTH: 50
    NORM: "SyncBN"
    STRIDE_IN_1X1: False
```

```
  FPN:
    NORM: "SyncBN"
  ROI_HEADS:
    NUM_CLASSES: 20
SOLVER:
  IMS_PER_BATCH: 16
  BASE_LR: 0.02
  STEPS: (18000, 22000)
  MAX_ITER: 24000
  WARMUP_ITERS: 100
TEST:
  PRECISE_BN:
    ENABLED: True
```

A.4 Fine-tuning tasks for Transfer: iNaturalist 2018

We use data augmentation and initialize the fully connected layer as done with ImageNet-1k linear classification (Appendix A.1). Despite a long-tailed distribution like LVIS, we do not perform any class resampling, following [25].

B Attention Visualizations for Predicted Caption

The *decoder attention module* in our textual head attends over a 7×7 grid of spatial image features and predicts a distribution of weights over these 49 candidate vectors. These spatial image features are then undergo weighted aggregation; and are used to predict the next token (in either direction). Having such an attention mechanism allows us to probe the model and check where it focuses in the image while making a prediction. We use our base model (ResNet-50 with $L = 1, H = 1024$) and predict the forward caption using beam search.

In Figures 5 and 6, we show attention masks overlaid on images corresponding to each time step, along with the predicted token. In Figure 7, we show attention masks overlaid on images corresponding to a randomly selected single token. The attention masks are a grid of 7×7 weights from decoder attention, upsampled to input image size by bicubic sampling. Note that the decoder attention is multi-headed (16 heads) – we average grids of attention weights produced by each head individually to form the final mask.