Notes on SwAV[1]

Seri Lee

January 7, 2021

[1] Caron, Mathilde, et al. "Unsupervised learning of visual features by contrasting cluster assignments." arXiv preprint arXiv:2006.09882 (2020).

Unsupervised image representations have significantly reduced the gap with supervised pretraining, notably with the recent achievements of contrastive learning methods. These contrastive methods typically work online and rely on a large number of explicit pairwise feature comparisons, which is computationally challenging. In the paper mentioned above, they propose an online algorithm, SwAV, that takes advantage of contrastive methods without requiring to compute pairwise comparisons.

Specifically, our method simultaneously clusters the data while enforcing consistency between cluster assignments produced for different augmentations (or "views") of the same image, instead of computing features directly as in contrastive learning.

Simply out, we use a "swapped" prediction mechanism where we predict the code of a view from the representation of another view.

Our method can be trained with large and small batches and scale to unlimited amounts of data. Compared to previous method, our method is more memory efficient since it does not require a large memory bank or a special momentum network.

In addition, we also propose a new data augmentation stratey, $multi-crop$, that uses a mix of views with different resolutions in place of two full-resolution views, without increasing the memory or compute requirements.

We validate our findings by achieving 75.3% top-1 accuracy on ImageNet with ResNet-50, as well as surpassing supervised pretraining on all the considered transfer tasks.

Method	Architecture	Parameter	top-1 accuracy
MoCo	ResNet50	24	60.6
PIRL	ResNet50	24	63.6
SIMCLR	ResNet50	24	70.0
SwAV	ResNet50	24	75.3

Table 1: **Linear classification on ImageNet**. Top-1 accuracy for linear models trained on frozen features from different self-supervised methods.

Introduction

Many recent state-of-the-art methods build upon the instance discrimination task that considers each image of the dataset ("instance") and its transformations as a separate class. This task yields representations that are able to discriminate between different images, while achieving some invariance to image transformations. Recent self-supervised methods that use instance discrimination rely

on a combination of two elements: (1) a contrastive loss and (2) a set of image transformations.

The contrastive loss removes the notion of instance classes by direcly mapping image features while the image transformations define the invariances encoded in the features. Both elements are essential to the quality of the resulting network.

The contrastive loss explicitly compares pairs of image representations to push away representations from different images while pulling together those transformations, or views, of the same image.

Since computing all the pairwise comparisons on a large dataset is not practical, most implementations approximate the loss by reducing the number of comparisons to random subsets of images during training. An alternative to approximate the loss is to approximate the task-that is to relax the instance discrimination problem. For example, clustering-based methods discriminate between groups of images with similar features instead of individual images.

The objective in clustering is tractable, but it does not scale well with the dataset as it requires a pass over the entire dataset to form image "codes" (*i.e.*, cluster assignments) that are used as targets during training. In this work, we use a different paradigm and propose to compute the codes online while enforcing consistency between codes obtained from views of the same image.

Comparing cluster assignments allows to contrast different image views while not relying on explicit pairwise feature comparisons. Specifically, we propose a simple "swapped" prediction problem where we predict the code of a view from the representation of another view.

We learn features by Swapping Assignments between multiples Views of the Same Image (SwAV). The features and codes are learned online, allowing our method to scale to potentially unlimited amounts of data.

Besides our online clustering-based method, we also propose an improvement to the image transformations. Most contrastive methods compare one pair of transformations per image, even though there is evidence that comparing more views during training improves the resulting model. In this work, we propose $multi-crop$ that uses smaller-sized images to increase the number of views while not increasing the memory or computational requirements during training. We also observe that mapping small parts of a scene to more global views significantly boosts the performance.

Directly working with downsized images introduces a bias in the features, which can be avoided by using a mix of different sizes. Our strategy is simple, yet effective, and can be applied to many self-supervised methods with consistent gain in performance.

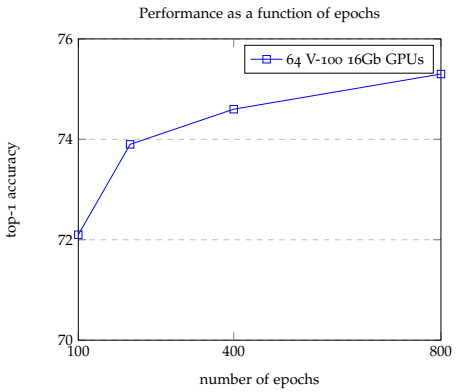

Figure 1: **Performance as a function of epochs**.Comparing SwAV models trained with different number of epochs and reporting their running time.

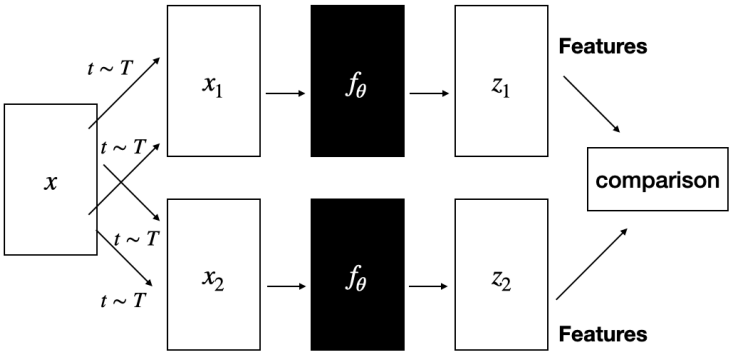

Figure 2: Contrastive instance learning. *the features from different transformations of the same images are compared directly to each other.*

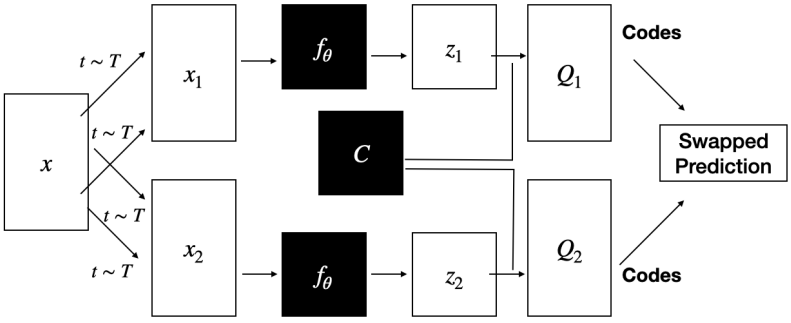

Figure 3: SwAV. *We first obtain "codes" by assigning features to prototype vectors. Then solve a "swapped" prediction problem wherein the codes obtained from one data augmented view are predicted using the other view. Prototype vectors are learned along with the ConvNet parameters by backpropagation.*

They validate their contributions by evaluating their protocol on several standard self-supervised benchmarks. In particular, on the ImageNet linear evaluation protocol, they reach 75.3% top-1 accuracy with a standard ResNet-50, and 78.5% with a wider model.

Method

Our goal is to learn visual

References

References are placed alongside their citations as sidenotes, as well. This can be accomplished using the normal `\cite` command.[2]

The complete list of references may also be printed automatically by using the `\bibliography` command. (See the end of this document for an example.) If you do not want to print a bibliography at the end of your document, use the `\nobibliography` command in its place.

To enter multiple citations at one location,[3] you can provide a list of keys separated by commas and the same optional vertical offset argument: `\cite{Tufte2006,Tufte1990}`.

`\cite[`⟨*offset*⟩`]{`*bibkey1*,*bibkey2*,...`}`

[2] The first paragraph of this document includes a citation.

[3] ; and

Figures and Tables

Images and graphics play an integral role in Tufte's work. In addition to the standard figure and tabular environments, this style provides special figure and table environments for full-width floats.

Full page–width figures and tables may be placed in figure* or table* environments. To place figures or tables in the margin, use the marginfigure or margintable environments as follows (see figure 4):

```
\begin{marginfigure}
  \includegraphics{helix}
  \caption{This is a margin figure.}
\end{marginfigure}
```

Figure 4: This is a margin figure. The helix is defined by $x = \cos(2\pi z)$, $y = \sin(2\pi z)$, and $z = [0, 2.7]$. The figure was drawn using Asymptote (`http://asymptote.sf.net/`).

The marginfigure and margintable environments accept an optional parameter ⟨*offset*⟩ that adjusts the vertical position of the figure or table. See the "**??**" section above for examples. The specifications are:

```
\begin{marginfigure}[⟨offset⟩]
    ...
\end{marginfigure}

\begin{margintable}[⟨offset⟩]
    ...
\end{margintable}
```

Figure 5 is an example of the `figure*` environment and figure 6 is an example of the normal `figure` environment.

Figure 5: This graph shows $y = \sin x$ from about $x = [-10, 10]$. *Notice that this figure takes up the full page width.*

Table 2 shows table created with the `booktabs` package. Notice the lack of vertical rules—they serve only to clutter the table's data.

Margin	Length
Paper width	8 1/2 inches
Paper height	11 inches
Textblock width	6 1/2 inches
Textblock/sidenote gutter	3/8 inches
Sidenote width	2 inches

Table 2: Here are the dimensions of the various margins used in the Tufte-handout class.

Full-width text blocks

In addition to the new float types, there is a fullwidth environment that stretches across the main text block and the sidenotes area.

```
\begin{fullwidth}
Lorem ipsum dolor sit amet...
```

Figure 6: Hilbert curves of various degrees n. *Notice that this figure only takes up the main textblock width.*

```
\end{fullwidth}
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Typography

Typefaces

If the Palatino, Helvetica, and `Bera Mono` typefaces are installed, this style will use them automatically. Otherwise, we'll fall back on the Computer Modern typefaces.

Letterspacing

This document class includes two new commands and some improvements on existing commands for letterspacing.

When setting strings of ALL CAPS or SMALL CAPS, the letterspacing—that is, the spacing between the letters—should be increased slightly.[4] The `\allcaps` command has proper letterspacing for strings of FULL CAPITAL LETTERS, and the `\smallcaps` command has letterspacing for SMALL CAPITAL LETTERS. These commands will also automatically convert the case of the text to

[4]

upper- or lowercase, respectively.

The `\textsc` command has also been redefined to include letterspacing. The case of the `\textsc` argument is left as is, however. This allows one to use both uppercase and lowercase letters: The Initial Letters Of The Words In This Sentence Are Capitalized.

Installation

To install the Tufte-LaTeX classes, simply drop the following files into the same directory as your `.tex` file:

```
tufte-book.cls
tufte-common.def
tufte-handout.cls
tufte.bst
```

More Documentation

For more documentation on the Tufte-LaTeX document classes (including commands not mentioned in this handout), please see the sample book.

Support

The website for the Tufte-LaTeX packages is located at `https://github.com/Tufte-LaTeX/tufte-latex`. On our website, you'll find links to our SVN repository, mailing lists, bug tracker, and documentation.