Published in Towards Data Science



KamWoh Ng Follow

Jun 27, 2021 · 9 min read ★ · Listen



# (Self-)Supervised Pre-training? Self-training? Which one to start with?

## What are the current state of the arts in self-supervised pre-training? Do we really need pre-training? How about self-training?

Recently, pre-training has been a hot topic in Computer Vision (and also NLP), especially one of the breakthroughs in NLP — BERT, which proposed a method to train an NLP model by using a "self-supervised" signal.

In short, we come up with an algorithm that can generate a "pseudo-label" itself (meaning a label that is true for a specific task), then we treat the learning task as a supervised learning task with the generated pseudo-label. It is commonly called "Pretext Task". For example, BERT uses mask word prediction to train the model (we can then say it is a pre-trained model after it is trained), then fine-tune the model with the task we want (usually called "Downstream Task"), e.g. review comment classification. The mask word prediction is to randomly mask a word in the sentence, and ask the model to predict what is that word given the sentence.

As a result, we can obtain a very good performance NLP model, by training it with a huge amount of **unlabelled** training data i.e., millions of sentences from the Internet (e.g. ),

sentence is erroneous and non-readable). Hence it is quite easy to define a pretext task in NLP. However, images are way more difficult as the signals are continuous, and pixels have a range of [0, 255], we as human are difficult to interpret a bunch of pixel values, and any shifting to the pixels will not be an issue to a human, but it will be completely different to a computer.

In this post, I will introduce what is a pre-trained model, downstream task, the current state of Self-Supervised Learning for Computer Vision, how to define a pretext task for the computer to learn **meaningful** and **invariant** features from images, and whether it is good to always apply pre-training, is there any alternative solution to pre-training. I assume readers have some basic understanding of CNN and Deep Learning.

#### The common flow of deep learning for computer vision project

Before talking about specific terms, let's look at the big picture of a deep learning project (specifically for computer vision project, but should apply to all others' project)

```
from torchvision.models import alexnet

model = alexnet(pretrained=True) # set pretrained=True
custom_task_model = prepare_custom_model()
dataset = load_dataset()
dataloader = DataLoader(dataset)

for epoch in range(epochs):
  for data, label in dataloader:
    # representation could be feature maps or vector
    representation = model(data)
    # compute specific outputs such as object detection outputs
    output = custom_task_model(representation)
    # compute specific task loss
    loss = criterion(output, label)
    loss.backward()
    # update model and/or custom_task_model
    optimizer.step()
```

For all almost open-source research projects, you will see the above pseudo-code. The

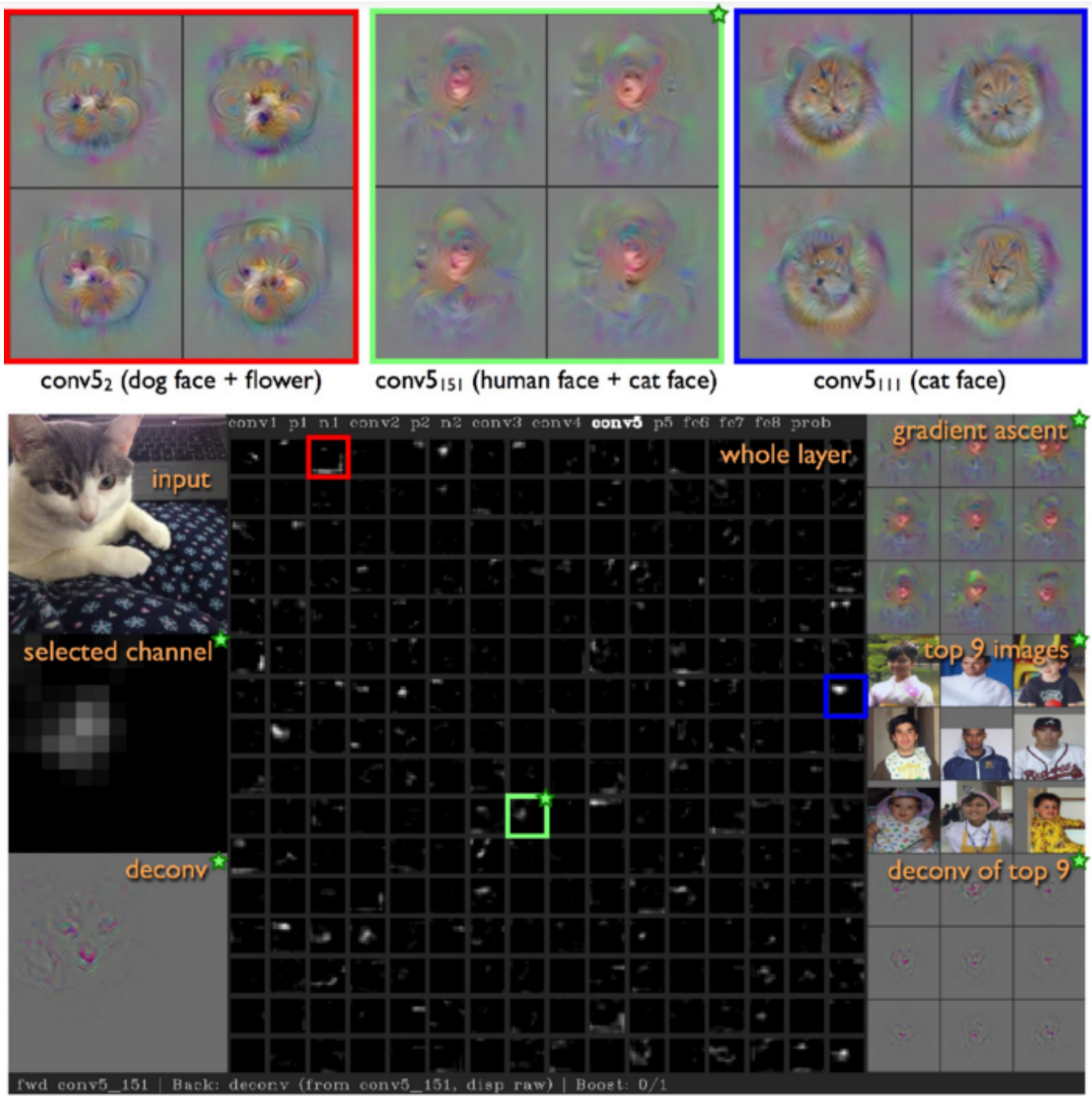
As you have noticed from the above pseudo-code, the “representation” is the most important part for any specific tasks (e.g. classification task), there are a lot of names for it, e.g. “embedding”, “vector”, “feature”. It literally means that this vector of real values is the “representation” that describes the data, even though it is difficult to be interpreted by a human, but it is actually **meaningful** for a computer to understand the data. It is because we want the computer to classify the input data, but pixels of data are too complicated, hence we want to extract “features” from the pixels of data, that is, **to find out what combination of pixels is actually able to describe the data.** Hence, it is important to let the computer learn how to combine these pixels so that it can classify them.

#### Pre-trained Model

Pre-trained models which are provided by **torchvision.models** were trained on ImageNet1000 through supervised learning (cross-entropy loss). The simplest way to obtain the pre-trained model is to set the keyword “pretrained=True” when you construct any models from this package (see the pseudo code above).

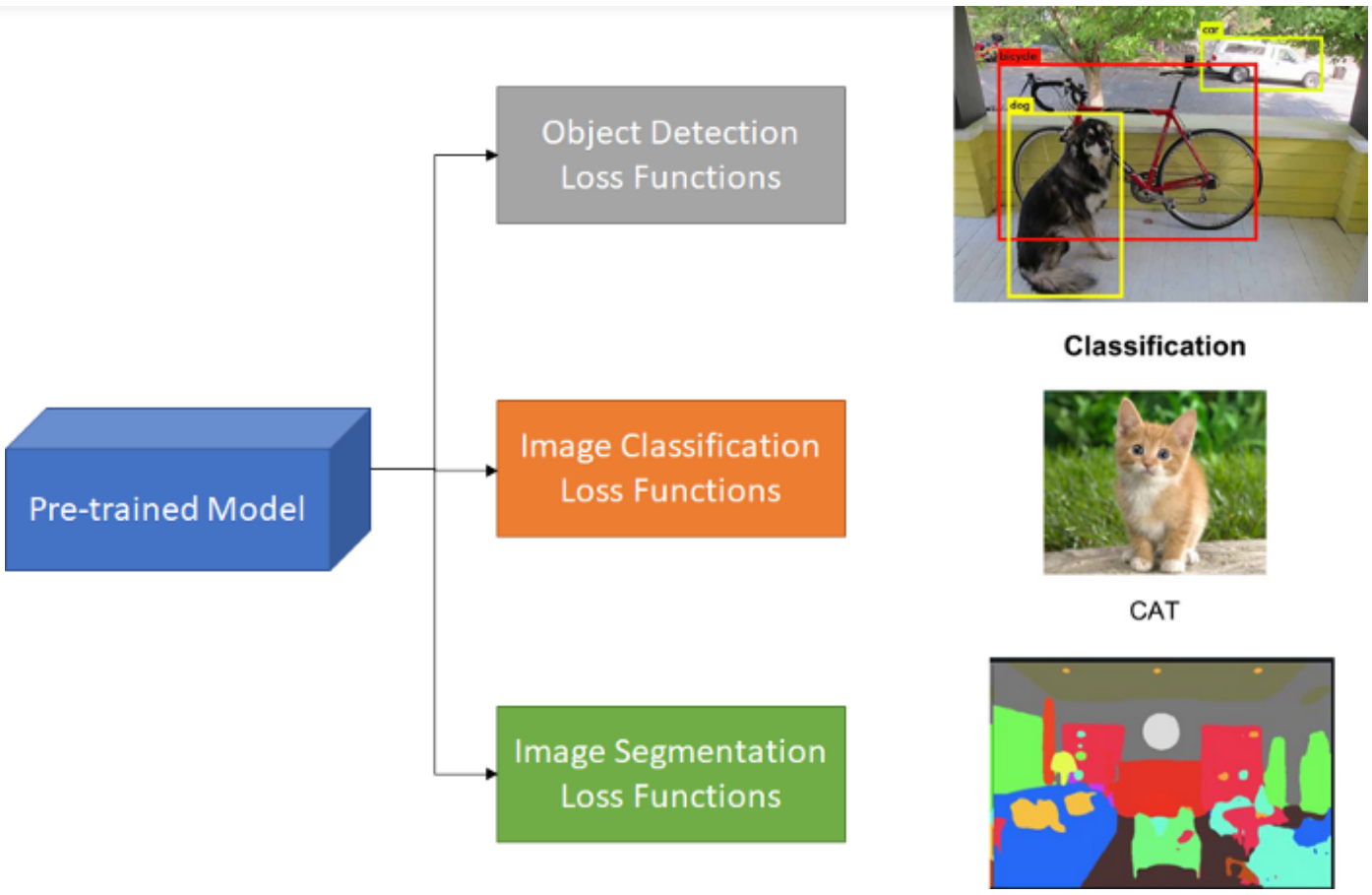
In the community of deep learning for computer vision, it is really common that we always begin with ImageNet pre-trained model, and fine-tune the model to a specific task with specific datasets. Because pre-trained model helps to save the time of training the model from scratch since the representation it has learned is already suitable (or easily to transfer) for a specific task such as Image Classification or Object Detection.




A figure from "Understanding Neural Networks Through Deep Visualization by Yosinki".

For example, the above figure is a deep visualization on a trained AlexNet. The feature maps (those with black background but with some white points) are the "activation" of some features, meaning that a specific channel in a Convolution layer has found *how to combine the pixels to represent some meaningful contents*, (e.g. the cat head). All pre-




An edited Image by the author. (With Credit: YOLO, ImageNet and CSAILVision)

When research papers mention “downstream task” (which confused me when I was a beginner), it literally means what the pre-trained model can do with image classification, object detection, image segmentation, etc. All these tasks are called “downstream tasks”, which is actually a technical term that is used by most representation learning papers.

In short, we fine-tune the model to perform specific tasks with the pre-trained model weights as initialization. Because some domains (such as the medical field) are very difficult and expensive to obtain data, so it is important to initialize with the pre-trained model, and also it will reduce the training time if you have computation constraints.

#### List of Self-Supervised Learning method that I know (by June 2021)

As far as I know, SSL can be organized as handcrafted pretext tasks-based, contrastive learning-based, clustering learning-based. Note that some papers will mention self-

takes as input an image), but I did not list them down here because some of them are too specific on the tasks, hence I think it is not good enough to be a pre-trained model.

#### Handcrafted Pretext Tasks

Some researchers propose to let the model learn to classify a human-designed task that does not need labeled data, but we can utilize the data to generate labels.

Context Prediction (predict location relationship)
Jigsaw
Predict Rotation
Colorization
Image Inpainting (learn to fill up an empty space in an image)

To summarize, these tasks use algorithms to generate pseudo-labels, so that model can learn representation through supervised learning such as cross-entropy loss. For the details algorithm, you can read the paper which I have put a link in the list.

#### Contrastive learning

The term “contrastive” means to differentiate, hence contrastive learning means learning to compare between positive and negative samples.

SimCLR
SimCLRv2
Moco
Mocov2
PIRL

These methods are sometimes called “instance discrimination” as well because they utilize the features of instances for learning. In short, the core concept is **to maximize the dot product between the feature vectors which are similar and minimize the dot product between those of which are not similar.** These methods have their own method to define similarity and dissimilarity. And most of them utilize “data augmentation”, that is, a positive pair is the same image with different kinds of

These methods are the current trend of self-supervised learning because these papers claim that contrastive learning can learn features that are more **invariant** (e.g. the feature map of a cat head I shown earlier, the location of the cat head in the image doesn't matter, it will activate as long as the cat head presents) as it can "compare" and find out what are the common features in the two augmented images.

#### Clustering learning

I put nearest neighbor-based as clustering learning as I view every data point as a center and finding the nearest neighbor as a cluster.

DeepClustering
SeLA
SCAN
SwAV

These methods are easier to understand, I use DeepClustering for an explanation as I think it is simple. We first perform clustering on the features (the number of centroids is a hyperparameter), then using the predicted labels from the clustering result as pseudo-labels, then treat it as an image classification problem and learn with cross-entropy loss. Again, different methods have their own version of finding pseudo-labels.

I personally recommend clustering-based learning, as it is more intuitive, but I personally think that two of the big problems are how to perform online clustering efficiently and accurately (meaning no need to go through all the data, but only the batch itself) and how to define the number of centroids correctly. (Nevertheless, SwAV said, as long as the number of centroids is "enough", the learning shouldn't be a problem).

#### Rethinking Pre-training and Self-Training

Well, the title is actually the same as a paper called "Rethinking Pre-training and Self-training". This paper finds out that pre-training is not as good as self-training in some cases.

To summarize, a self-training method has common steps as below:




3. retrain a (student) model with both labeled data and pseudo-labeled unlabeled data.

4. repeats steps 1 2 and 3. (by initializing the teacher model with the student model)

It is actually a semi-supervised learning method. The term “self” means that the model learns with some data first (and the model is initialized randomly), then using its own knowledge to classify new unseen data, and uses the highly confident prediction result as new data and learns with them. Hence, the term “self” means the model is teaching itself.

This paper said that although self-training is much slower than pre-training, it actually has many benefits such as **able to learn more specific features for the specific task**, **works well even when pre-training fails** and **it works even well with more data**. In the paper they demonstrate the benefit of self-training with object detection and semantic segmentation, it actually makes sense as the features from the pre-trained model were learned from classification task, but adapting it to localization task (i.e., object detection) might take time to adjust (or maybe not able to adjust accurately due to local minimum). While training from scratch will help as the features are tuned from random to the specific localization task, just that it takes more time for the training.

Nevertheless, I still think pre-training is good in practice because we can save a lot of time training the model, especially for fast demo or model deployment. But you can give self-training a try if you want to beat SOTA in your research field.

#### Conclusion

Hope this post can help you to understand more about the pre-training model, self-supervised learning methods, and also able to explore a new field that is called “self-training” (although it is not new to the community, it is new to me).

Auto-encoder-based can also consider as self-supervised learning, as it is reconstructing the input (the label is the input), but some papers said that auto-encoder (reconstructive based to be specific) will try to memorize every detail of the input, hence it is not “invariant” enough. It is good to have a **discriminative** feature, even if we lose information, as long as the information that is important to the main content in the image




If you wish to know more about self-supervised learning, I recommend you to read these surveys, 1, 2, 3. You can also search in Google Scholar with the keyword “self-supervised learning survey”. There is also another paper called “How Useful is Self-Supervised Pretraining for Visual Tasks?”, explains how to self-supervised pretraining help in visual tasks, I recommend readers to read if you have interest in self-supervised learning.