Transfer learning and self-supervised learning

Szu-Chi Chung

Department of Applied Mathematics, National Sun Yat-sen University

Why use transfer/self-supervised learning?

- It is often easier to obtain unlabeled data - from a lab instrument or a computer - than labeled data, which can require human intervention
  - Prepare labeling manuals, categories, hiring humans, creating GUIs, storage pipelines, etc.
- Sometimes it is also hard to label the data. For example it is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?
- Cognitive motivation: How animals / babies learn

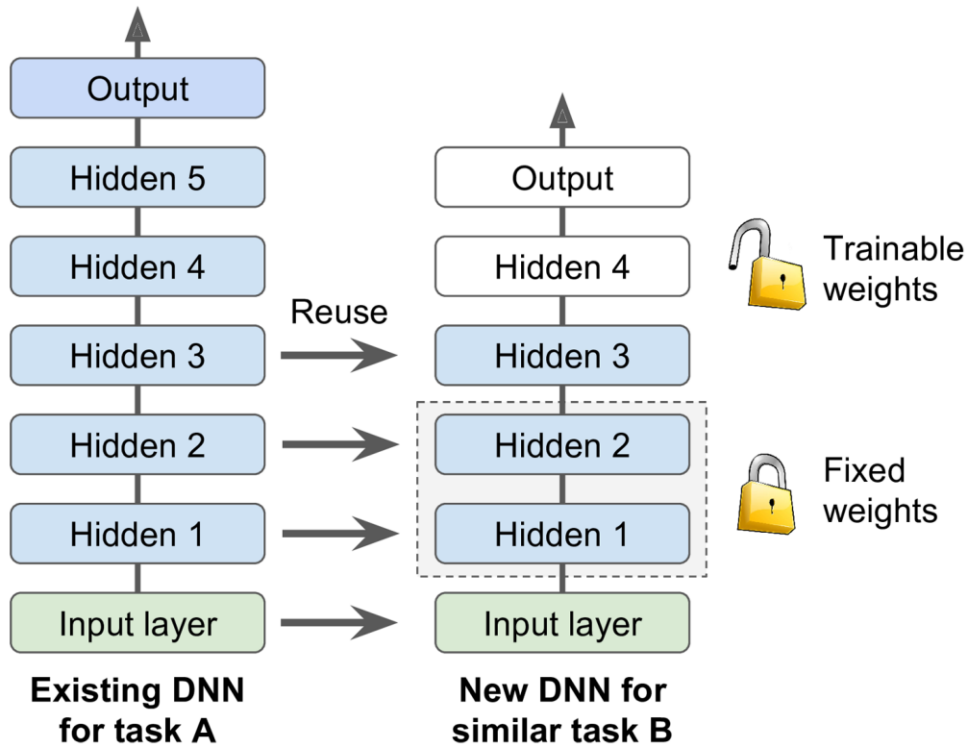
"The brain has about 10^{14}synapses and we only live for about 10^{9} seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning" - Geoffrey Hinton

1. Transfer learning

- It is generally not a good idea to train a very large DNN from scratch: instead, you should always try to find an existing neural network that accomplishes a *similar task* to the one you are trying to tackle, then reuse the lower layers of this network. This technique is called *transfer learning*
  - It will not only speed up training considerably, but also require significantly less labeled training data
- Suppose you have access to a DNN that was trained to classify pictures into 100 different categories, including animals, plants, vehicles, and everyday objects. You now want to train a DNN to classify specific types of vehicles. These tasks are very similar, even partly overlapping, so you should try to reuse parts of the first network

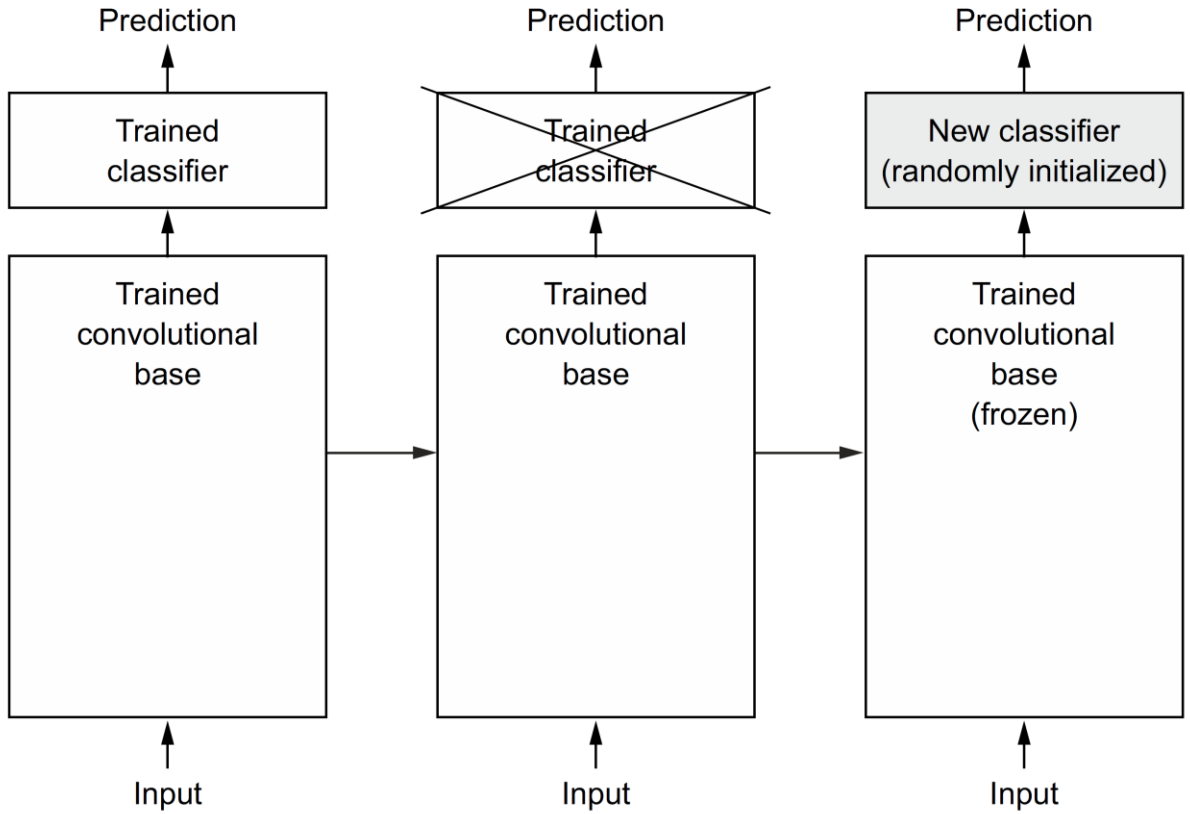
Transfer learning

- Transfer learning work best when the inputs have similar low-level features
  - The output layer of the original model should be replaced because it is most likely not useful for the new task, and it may not have the right number of outputs for the new task
  - Similarly, the upper hidden layers of the model are less likely to be as useful as the lower layers since the high-level features that are most useful for the new task may differ significantly from the ones that were most useful for the original task
  - If we treat the lower layer frozen, we can often speed up training while still obtaining good performance



Using pretrained model – feature extraction

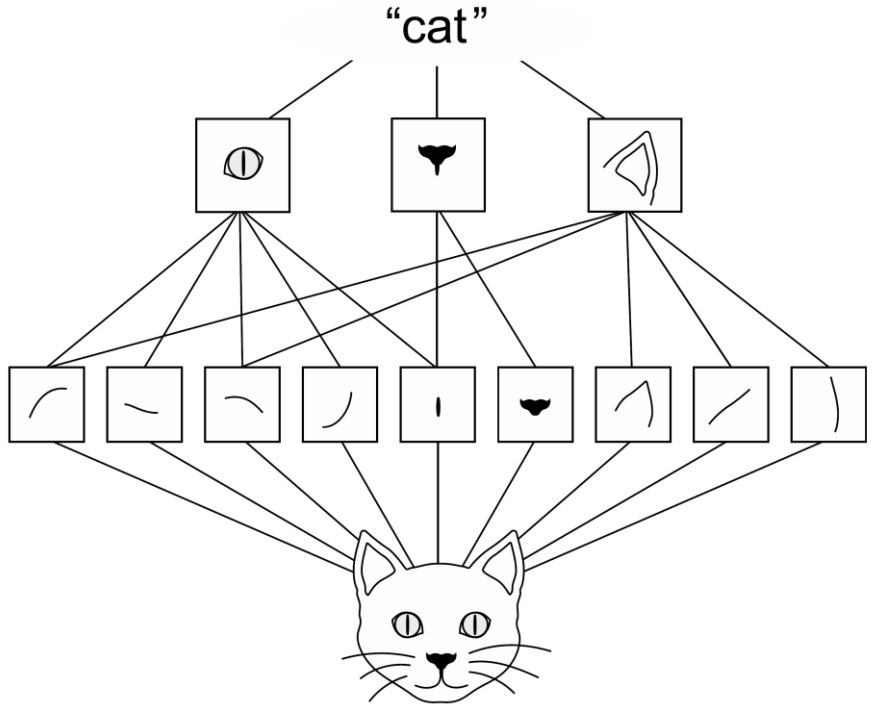
- *Feature extraction* with a pretrained model is often useful in the visual task
  - Such portability of learned features across different problems is an advantage of deep learning compared to traditional approaches and is effective for small-data problems



Using pretrained model – feature extraction

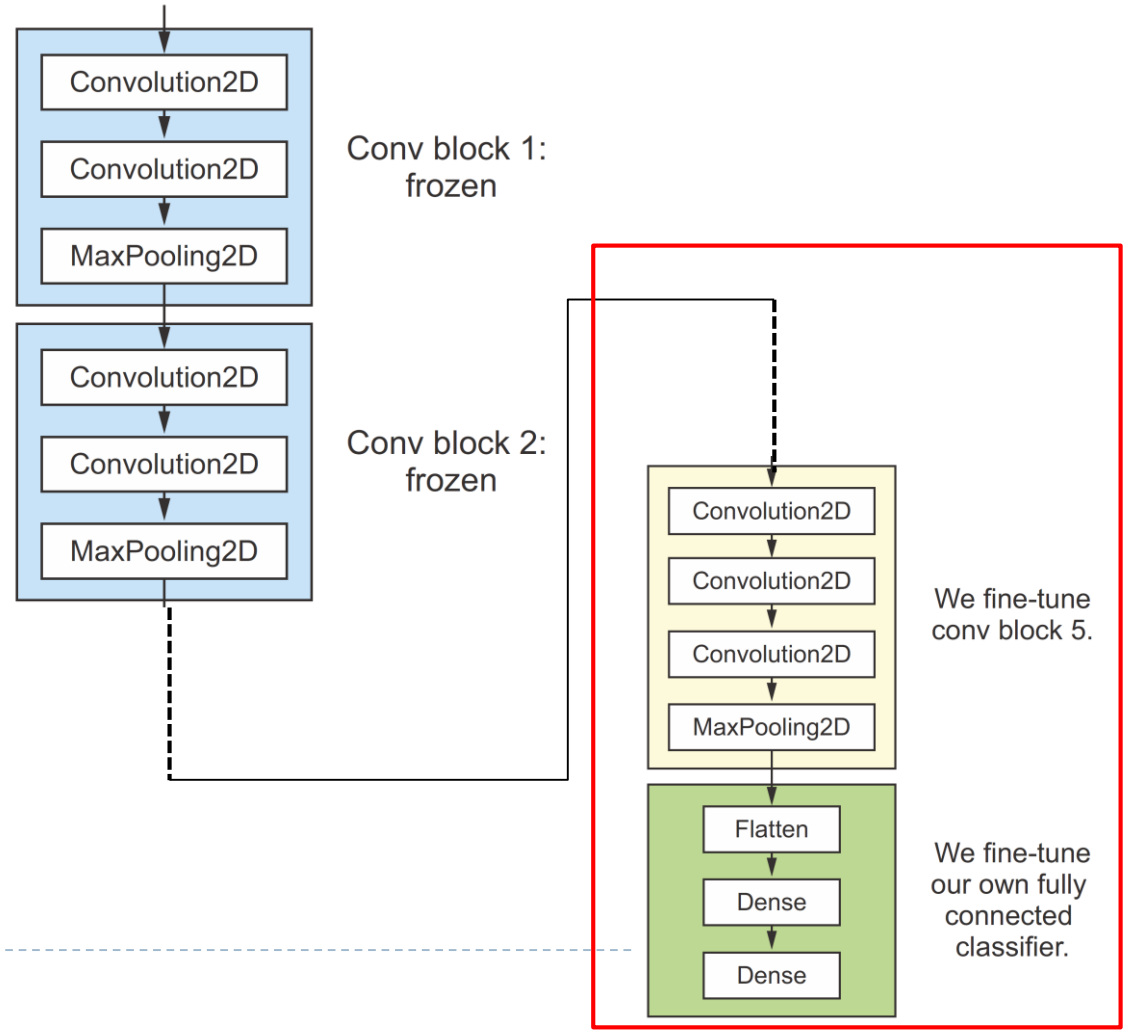
- Convnets start with series of pooling and convolution layers, and end with a densely connected classifier. The first part is *convolutional base* of the model
  - In the case of convnets, feature extraction consists of taking the convolutional base of a previously trained network, and training a new classifier on top of the output

- Note that the level of generality of the representations extracted by specific convolution layers depends on the depth of the layer
  - Layers that come earlier in the model extract local, highly generic feature maps (such as visual edges, colors, and textures), whereas layers that are higher up extract more-abstract concepts (such as “cat ear” or “dog eye”)



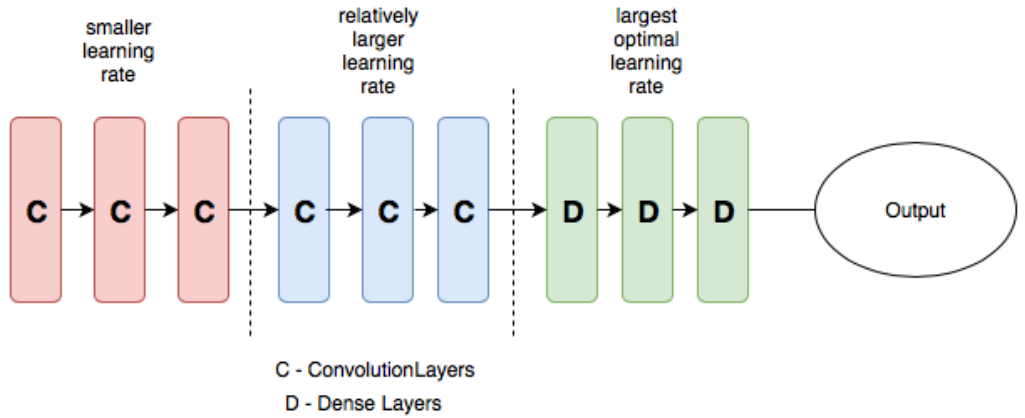
Using pretrained model – fine-tuning

- Another widely used technique for model reuse, complementary to feature extraction, is *fine-tuning*
  - Fine-tuning consists of unfreezing a few of the top layers of a frozen model base used for feature extraction, and jointly training both the newly added fully connected classifier and these top layers
  - When fine-tuning a model that includes BatchNormalization layers, it is sometimes recommended leaving these layers frozen



Discriminative learning rate

- Even after we unfreeze, we still care about the quality of pretrained weights
  1. We would not expect that the best learning rate for those pretrained parameters would be as high as for the randomly added parameters
  2. It often makes sense to let the later layers fine-tune more quickly than earlier layers
  3. Use a lower learning rate for the early layers of the neural network, and a higher learning rate for the later layers (and especially the randomly added layers)



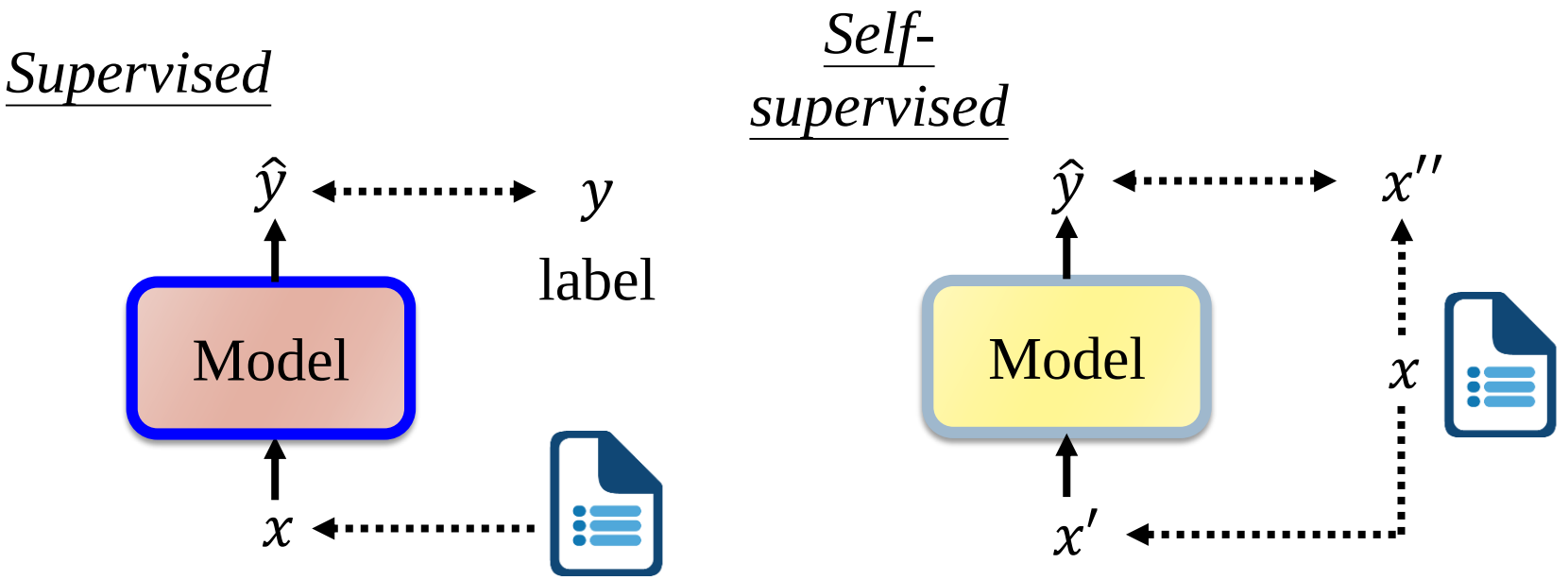
https://github.com/gunchagarg/differential-learning-rate-keras

From pretrained model to foundation model

- The pre-training task may be supervised or unsupervised; the main requirements are that it can teach the model's basic structure about the problem domain and that it is sufficiently similar to the downstream fine-tuning task
  - The notion of task similarity is not rigorously defined, but in practice, the domain of the pre-training task is often broader than that of the fine-tuning task
    - For example, it is common to use the ImageNet dataset to pretrain CNNs, which can then be used for a variety of downstream tasks. ImageNet has 1.28 million natural images, each associated with a label from one of 1,000 classes.
    - For NLP, we could pre-train a model on a large English-labeled corpus before fine-tuning on low-resource languages
  - A *foundation model*, also known as large AI model, is a machine learning or deep learning model that is trained on vast datasets so it can be applied across a wide range of use cases
  - How about going beyond supervised pretrain?

2. Self-supervised learning

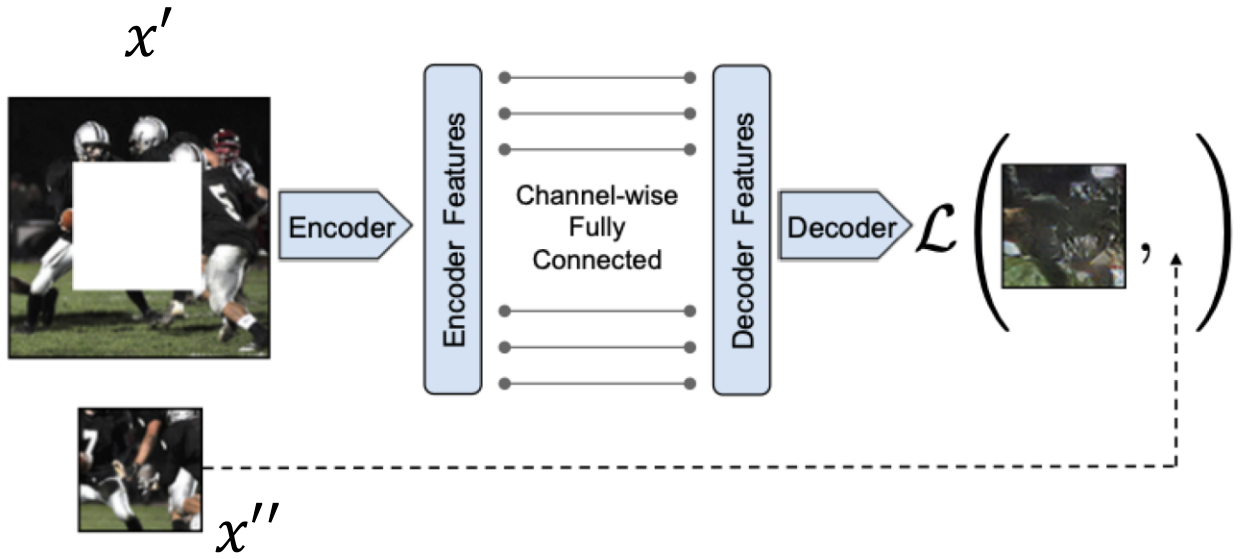
- Self-supervised learning is an active research field. Self-supervised learning is an approach to pre-training models *using unlabeled data*
  - The labels are created by the algorithm, rather than being provided externally by a human, as in standard supervised learning. Both supervised and self-supervised learning still require predicting outputs given inputs



https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/bert_v8.pptx

Self-supervised learning - Pretrained using imputation tasks

- One approach to self-supervised learning is to solve *imputation tasks*. In this approach, we partition the input vector x into two parts, $x = (x'', x')$, and then try to predict the hidden part x'' given the remaining visible part, x', using a model of the form $\hat{y} = f(x'' = 0, x')$



https://sites.google.com/view/berkeley-cs294-158-sp20/home

Self-supervised learning - Pretrained using imputation tasks

- For more information see here

(special token)

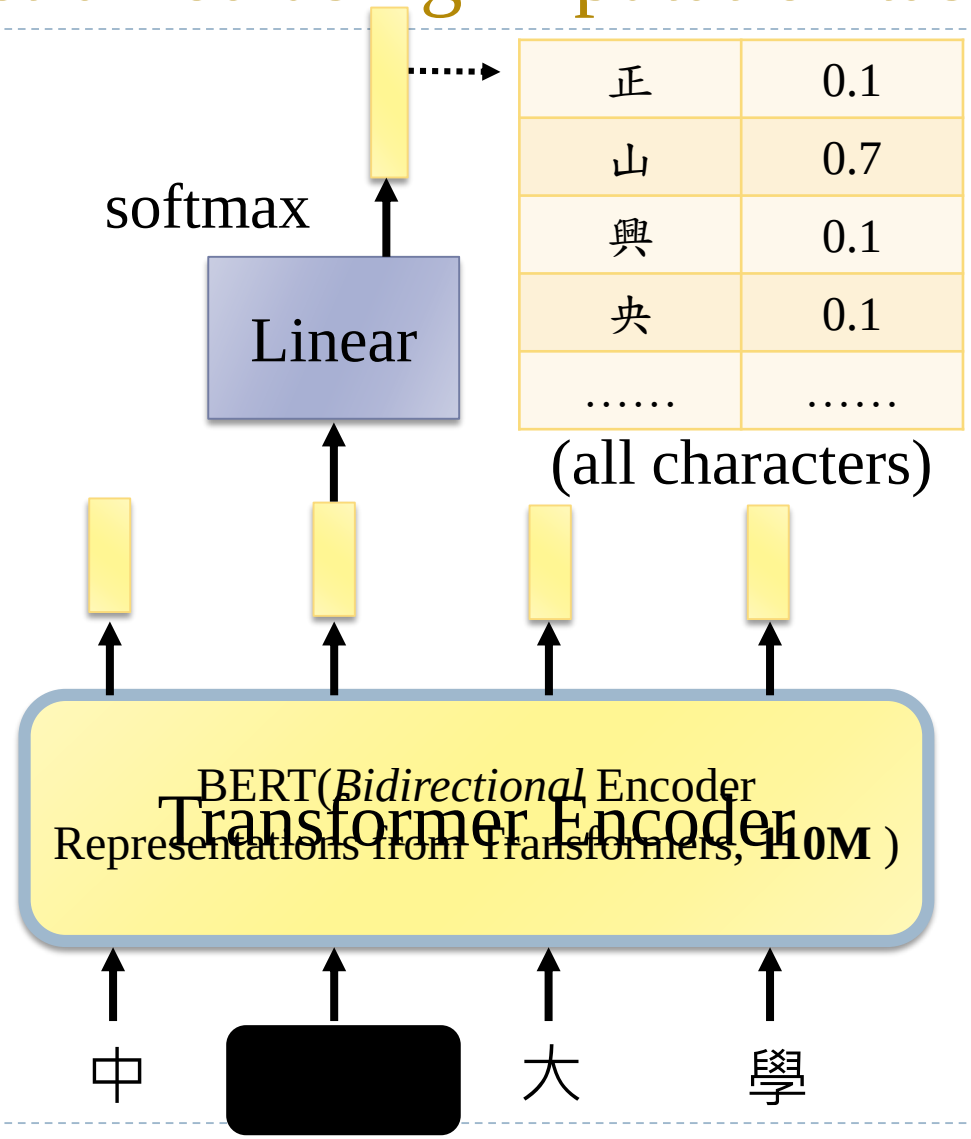
[redacted] = MASK

or

[redacted] = Random

一、二、大、小 …

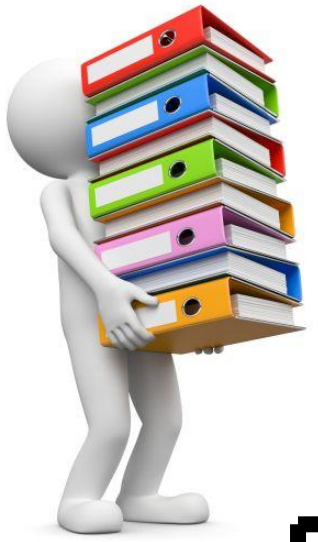
Randomly masking some tokens



https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/bert_v8.pptx

Downstream task - NLP

Data:
800M+2500M



Self-supervised Learning

Pre-train

BERT

- Masked token prediction

Fine-tune

Model for text classification

Model for sentiment analysis

Model for translation

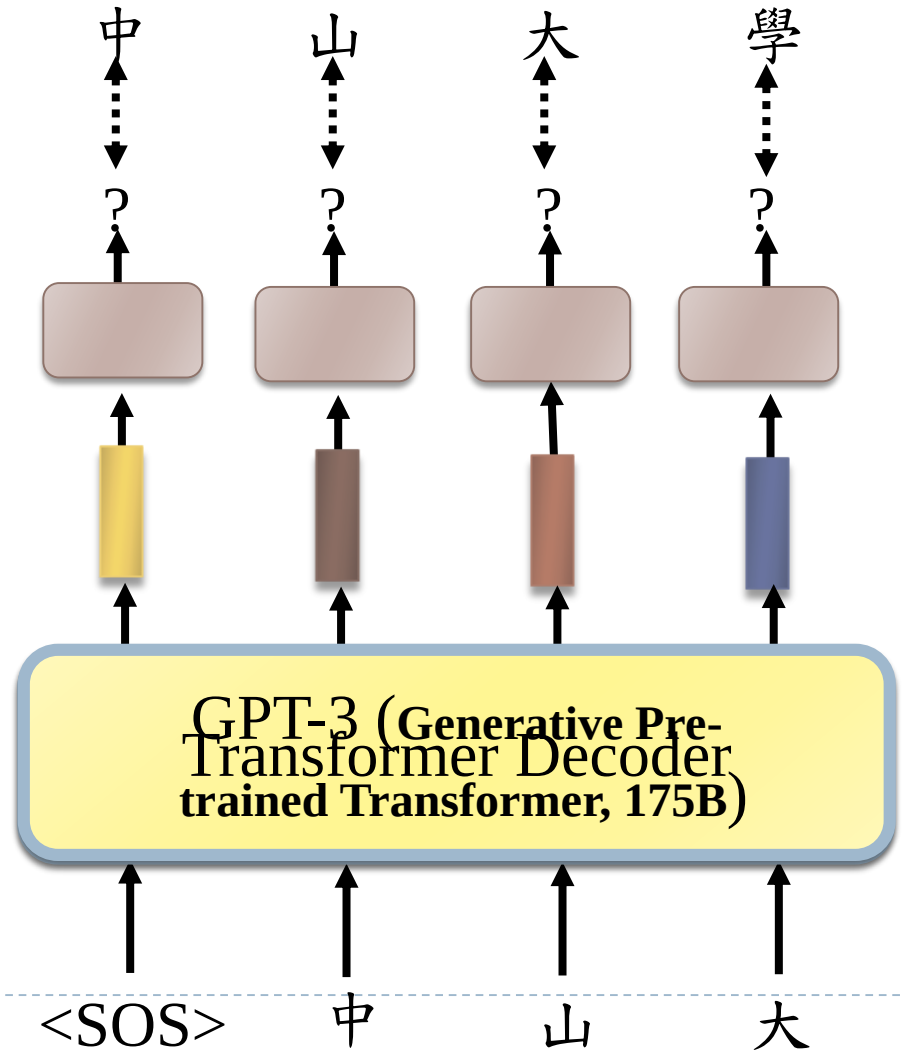
Downstream Tasks

- The tasks we care
- We have a little bit labeled data

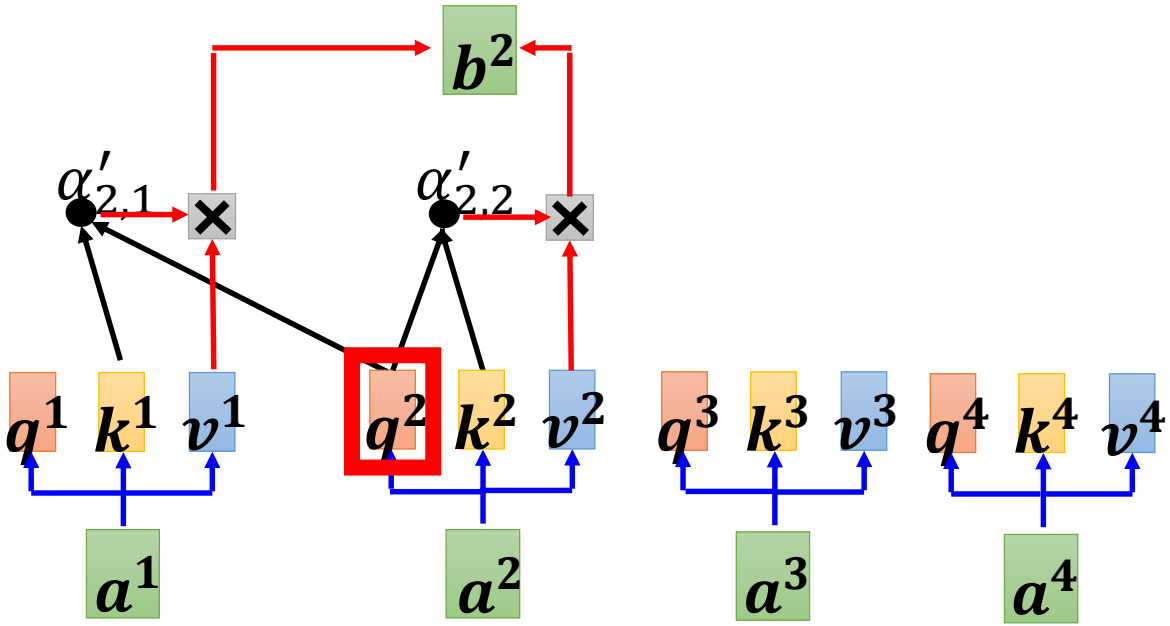
https://speech.ee.ntu.edu.tw/~hylee/ml/ml2021-course-data/bert_v8.pptx

Self-supervised learning - Pretrained using imputation tasks

- Predict Next Token

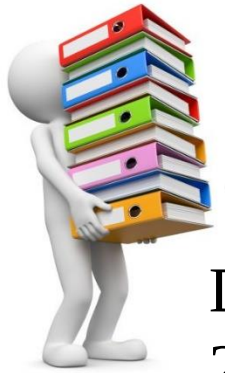


Masked attention



Downstream task - NLP

- In-context learning



Data:
200B~300B

GPT

"Few-shot" Learning

(no gradient descent)

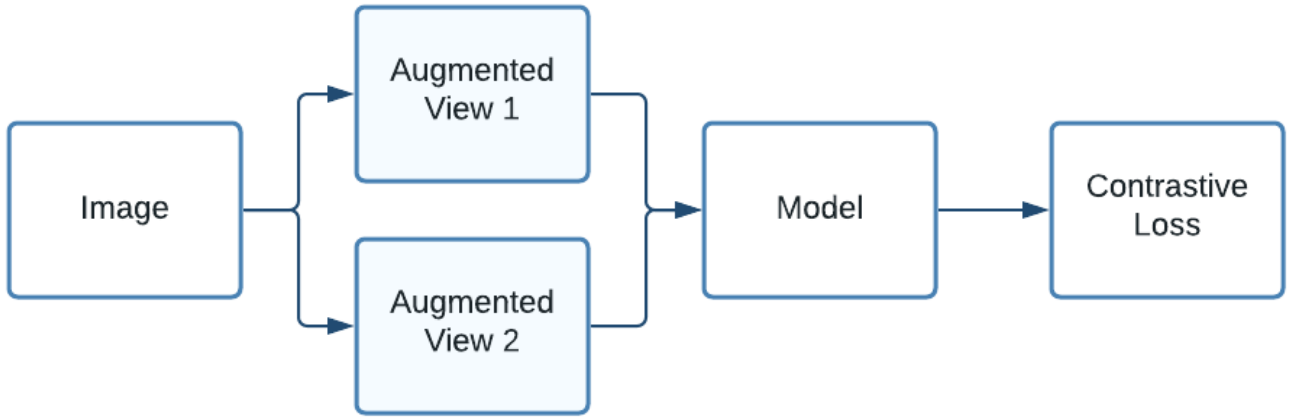
```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ← examples
4  plush girafe => girafe peluche      ← examples
5  cheese =>                           ← prompt
```

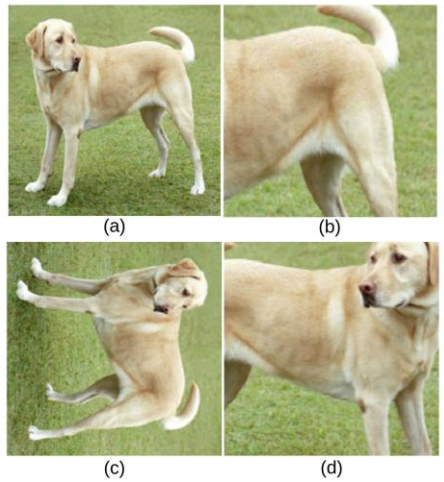
"zero-shot" Learning

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

3. Self-supervised learning - contrastive tasks

- Solve proxy tasks, also called *pretext tasks*
  - The basic idea is to create pairs of examples that are semantically similar to each other, using data augmentation methods
  - Train a self-supervised model to learn data representations by contrasting multiple augmented views of the same example. These learned representations capture data invariants, e.g., object translation, color jitter, noise, etc

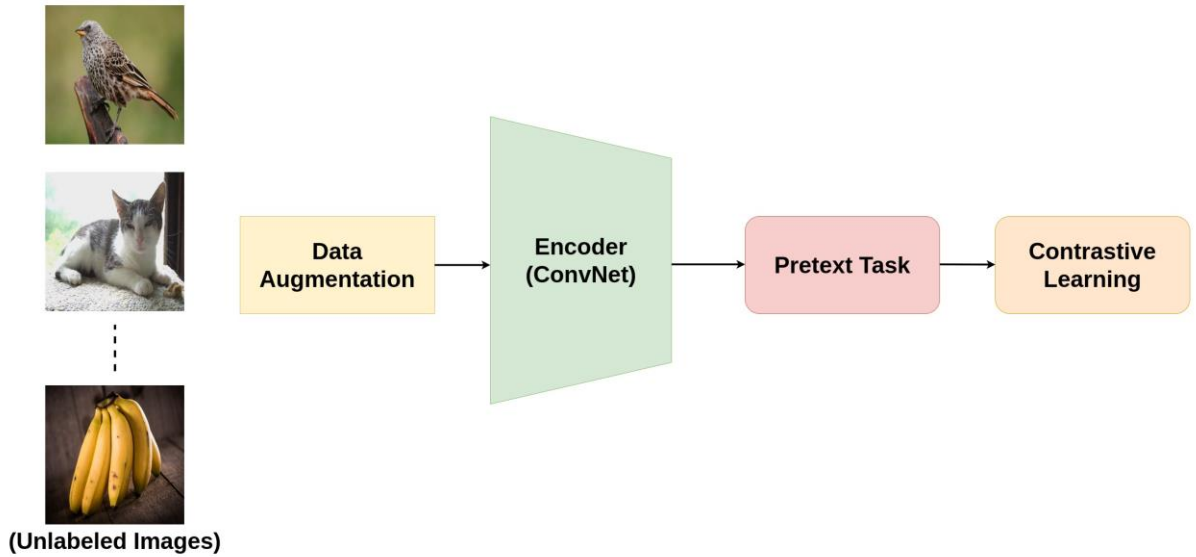


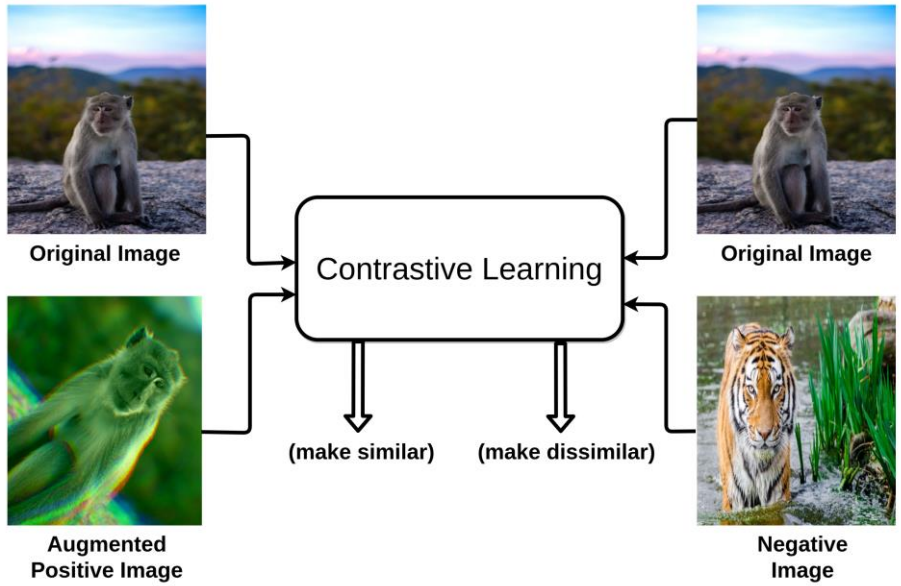


https://arxiv.org/abs/2011.00362

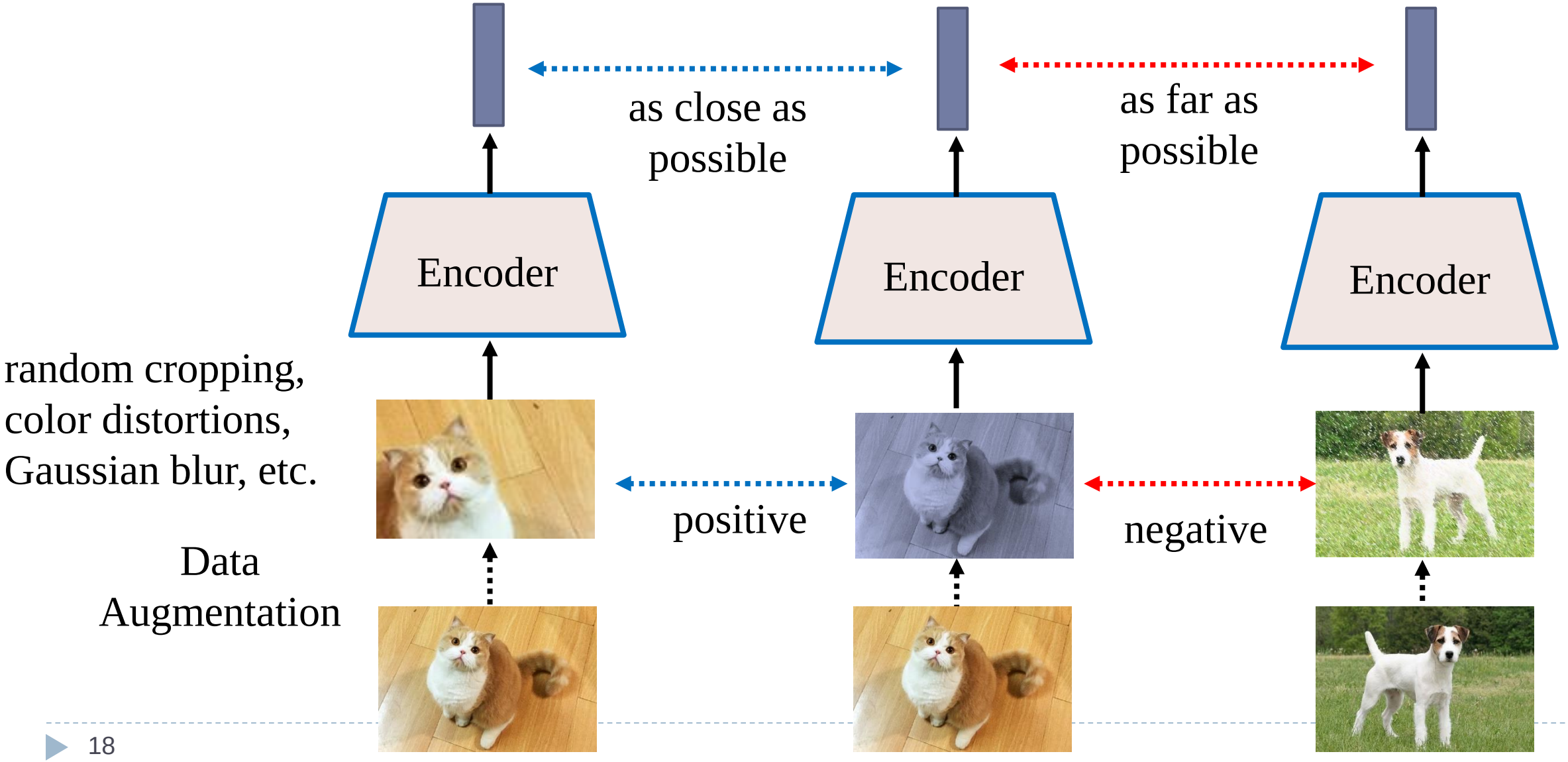
Self-supervised learning - contrastive tasks

- Pretext tasks are self-supervised tasks that act as an important strategy to learn representations of the data
  - The original image acts as an anchor, its augmented version acts as a positive sample, and the rest of the images in the batch or in the training data act as negative samples





Self-supervised learning - contrastive tasks - SimCLR



https://speech.ee.ntu.edu.tw/~hylee/ml/ml2022-course-data/SSL_speech_image%20(v9).pptx

Selecting Negative Examples is not trivial

- The negative examples should be hard enough

But cannot be too hard …



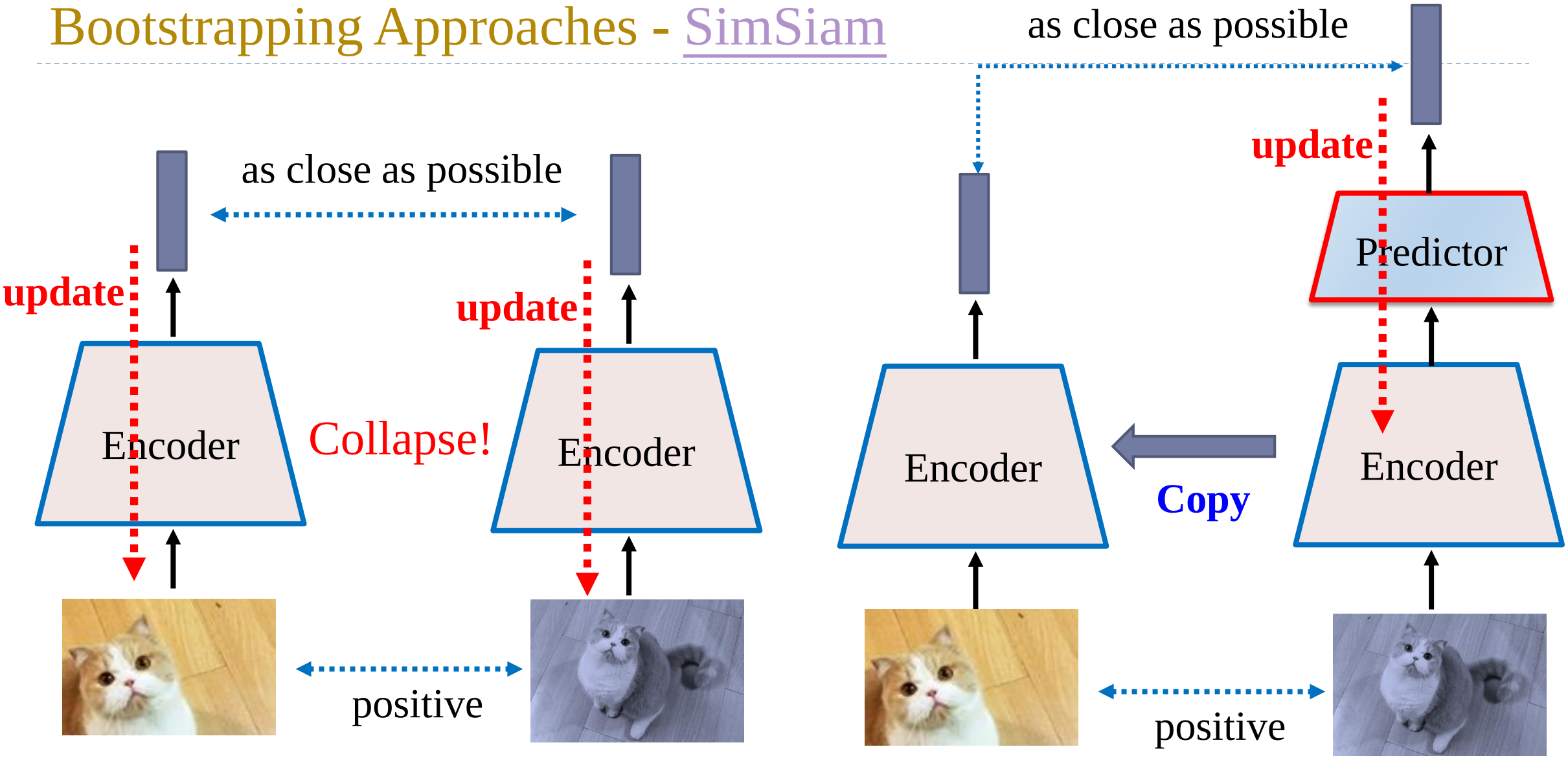






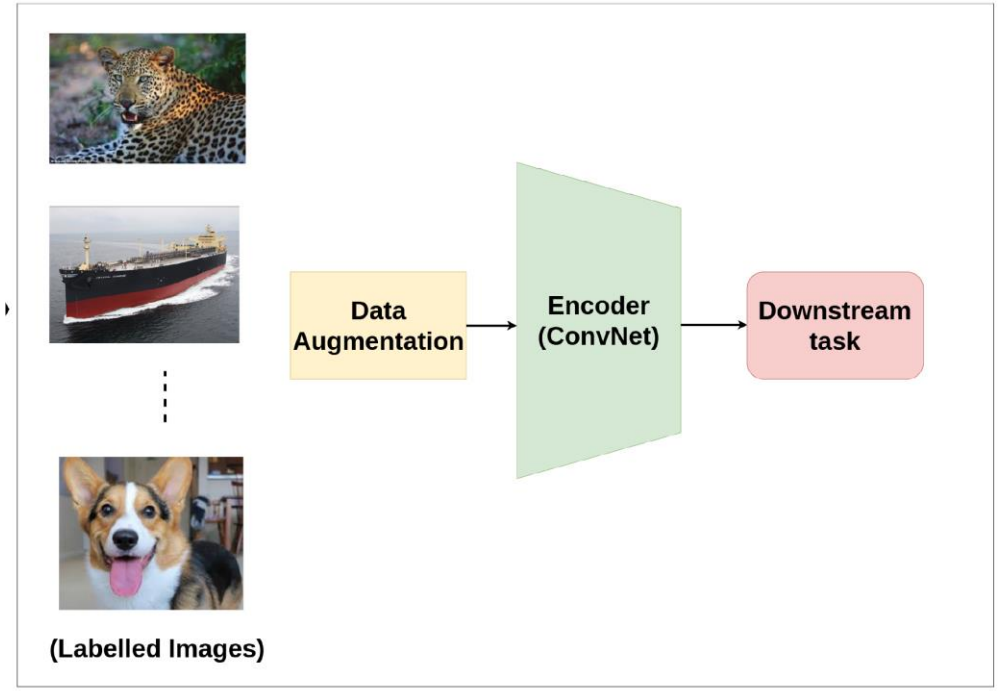



Bootstrapping Approaches - SimSiam



Downstream task – computer vision

- Downstream tasks are application-specific tasks that utilize the knowledge that was learned during the contrastive task
  - The learned parameters serve as a pretrained model and are transferred to other downstream computer vision tasks by fine-tuning
    - Training a classifier on top of the frozen representations
  - The encoder can then be used to produce embedding or latent space

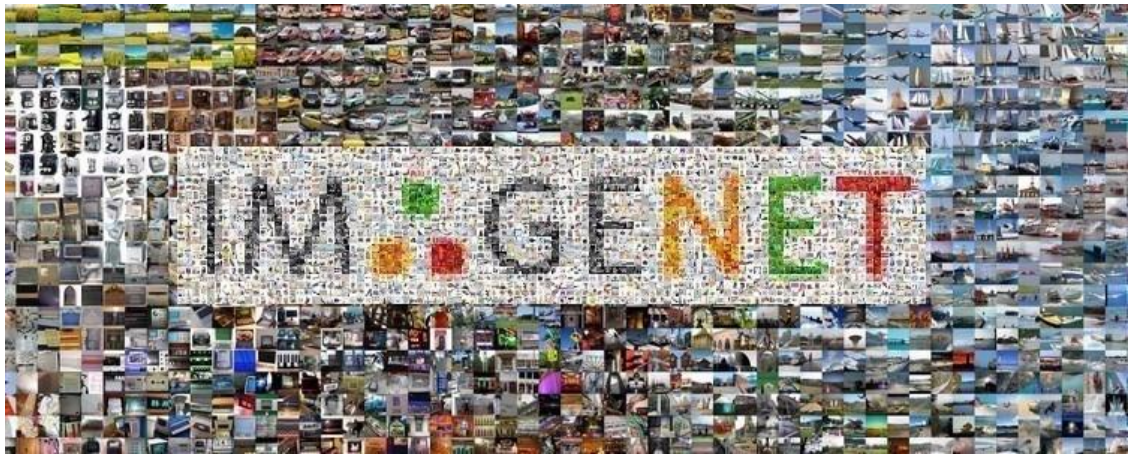


Multimodal - CLIP (Contrastive Language–Image Pre-training)

- Text
  - Self-supervised (LLM)
  - Large training data
- Images
  - Supervised learning
  - Not that large training data

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion

https://arxiv.org/abs/2203.15556

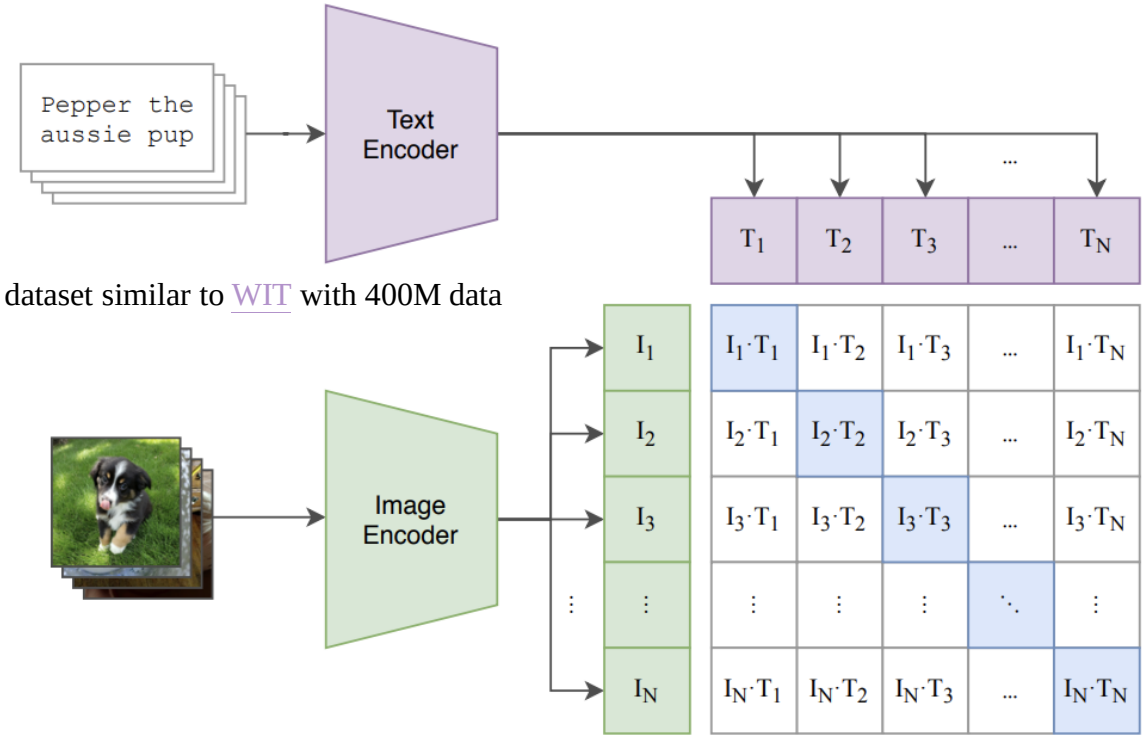


1.28M, 1000 classes

CLIP (Contrastive Language–Image Pre-training)

$$\min\left(\sum_{i=1}^{N}\sum_{j=1}^{N}(I_i \cdot T_j)_{i\neq j} - \sum_{i=1}^{N}(I_i \cdot T_i)\right)$$

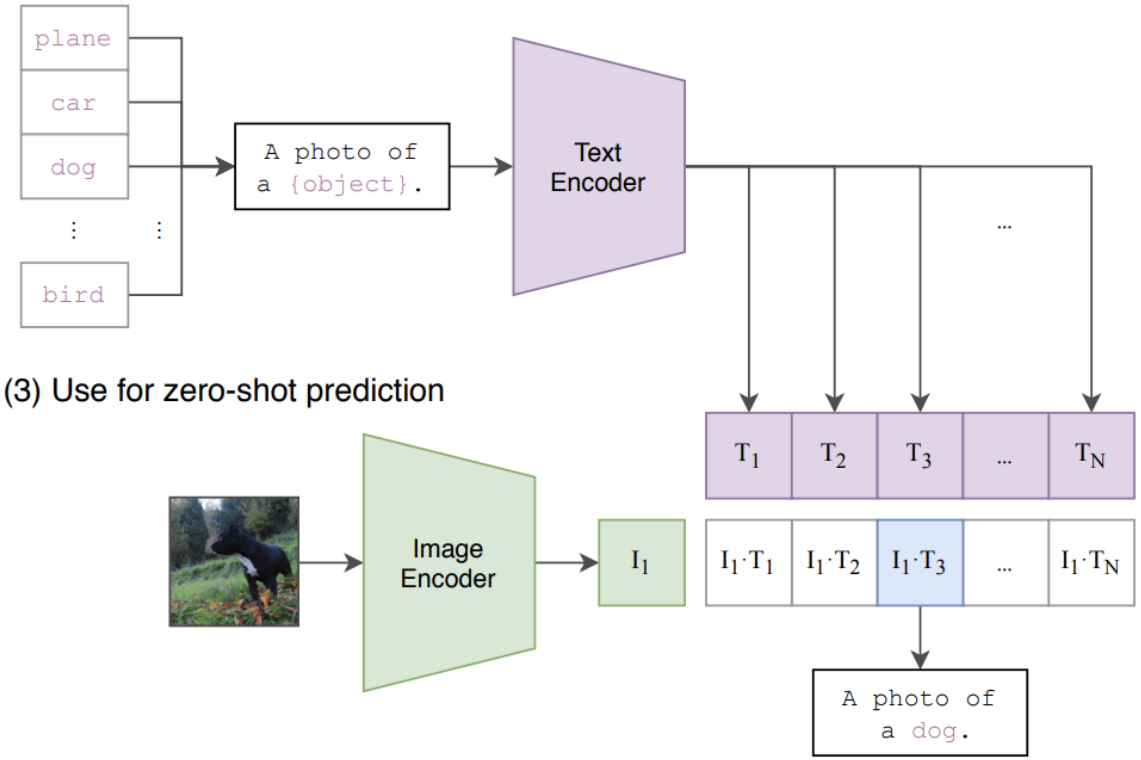
(1) Contrastive pre-training



Use dataset similar to WIT with 400M data

Training

(2) Create dataset classifier from label text

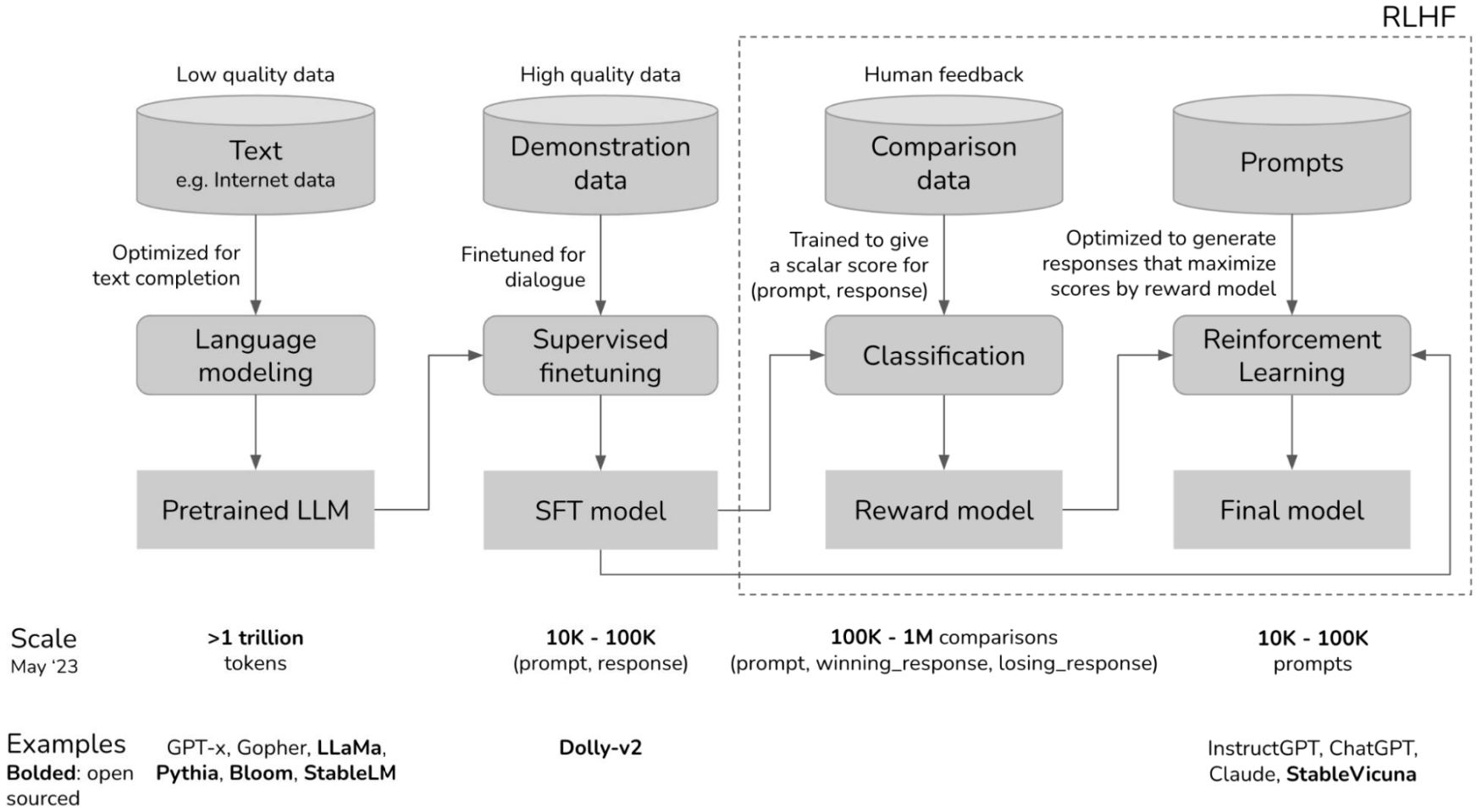


Inference

https://fullstackdeeplearning.com/course/2022/lecture-7-foundation-models/

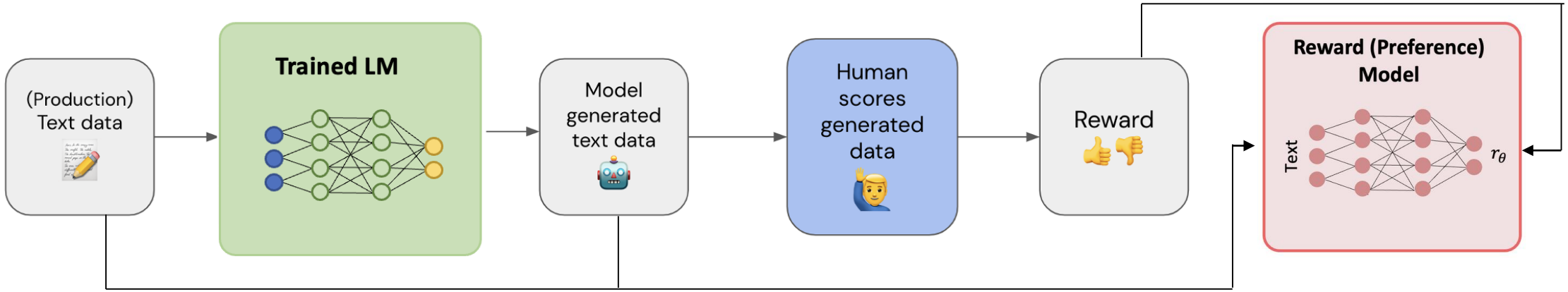
4. From GPT to ChatGPT

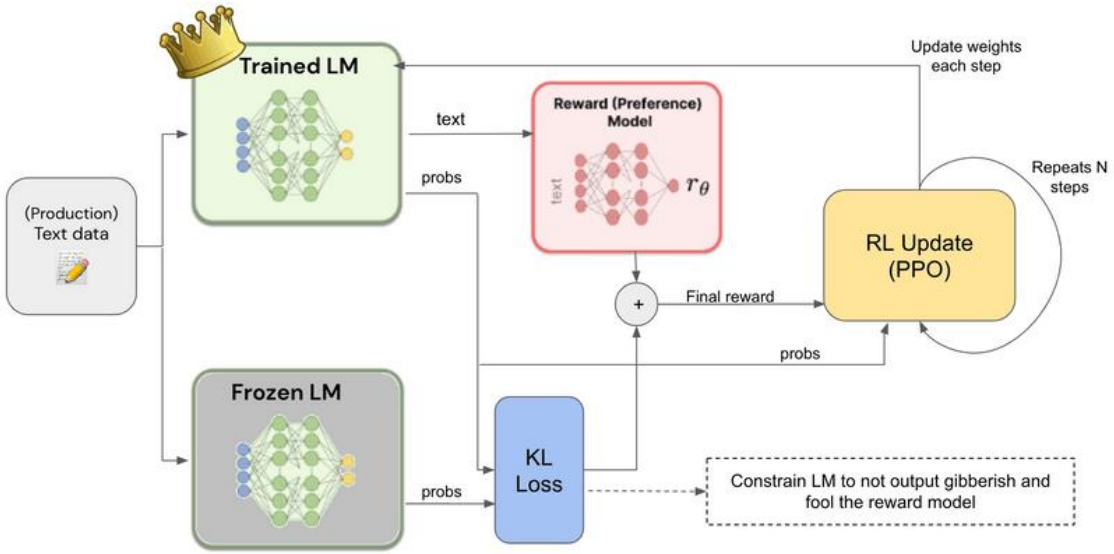
- Using Reinforcement Learning from Human Feedback (RLHF)



https://huyenchip.com/2023/05/02/rlhf.html

Reinforcement Learning from Human Feedback (RLHF)



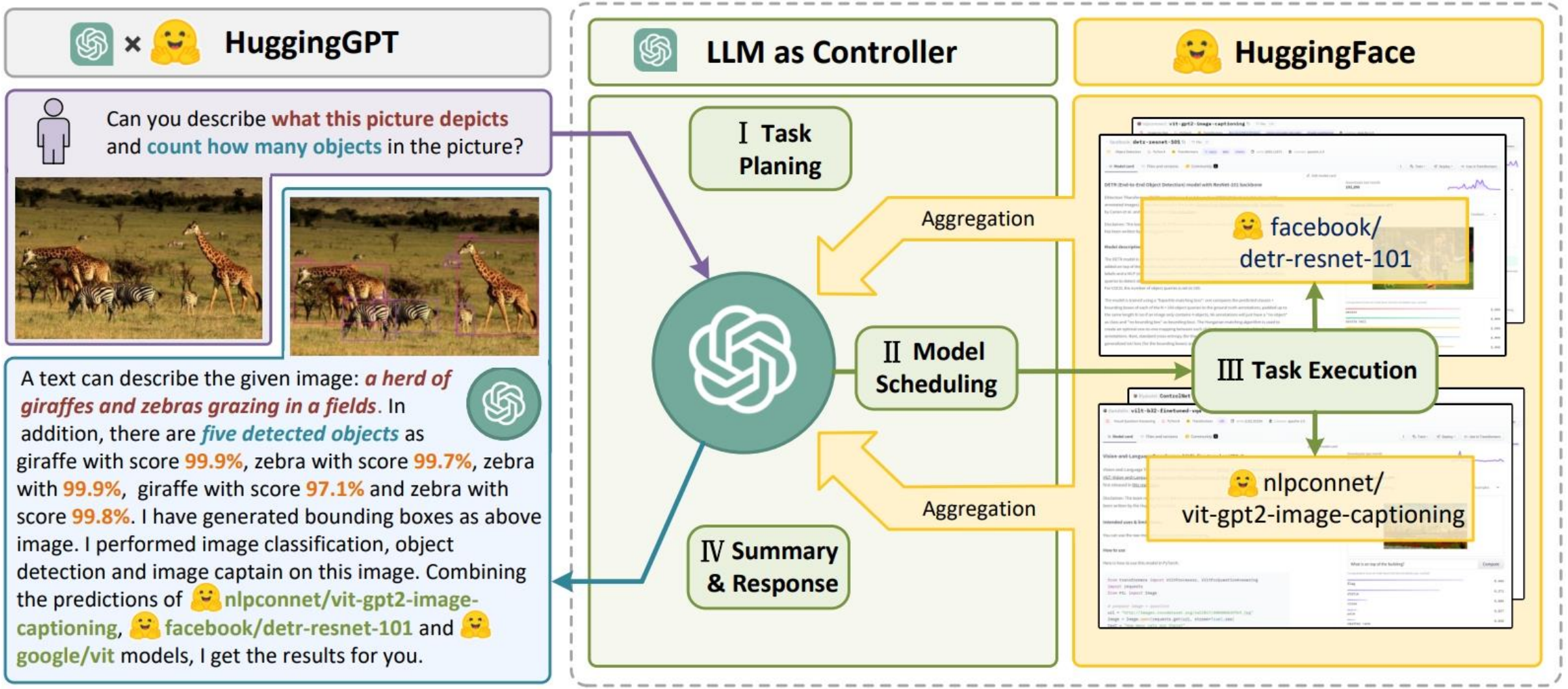


https://gist.github.com/JoaoLages/c6f2dfd13d2484aa8bb0b2d567fbf093

Recent advance

- HuggingChat
  - Based on LLaMA and finetune on an open dataset provided by OpenAssistant
- Dolly 2.0
  - Based on Pythia that can be used in commercial product
  - The dataset was authored by more than 5,000 Databricks employees during March and April of 2023
- HuggingGPT
  - Language serves as an interface for LLMs to connect numerous AI models for solving complicated AI tasks!

Recent advance



Conclusion

- Many ML models, especially neural networks, often have many more parameters than we have labeled training examples
  - Of course these parameters are highly correlated, so they are not independent “degrees of freedom”. Nevertheless, such big models are slow to train and, more importantly, they may easily overfit. This is particularly a problem when you do not have a large labeled training set
- Pretraining using supervised, unsupervised or self-supervised way can greatly benefit the downstream tasks by transferring knowledge from one task to another

References

[1] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition Chapter 11,14

[2] Deep learning with Python, 2nd Edition Chapter 8

[3] https://speech.ee.ntu.edu.tw/~hylee/ml/2021-spring.php

[4] https://speech.ee.ntu.edu.tw/~hylee/ml/2022-spring.php Lecture 7, 11

[5] A Survey on Contrastive Self-supervised Learning

[6] https://sites.google.com/view/nlpcolearningnsysu/events?authuser=0

Appendix

Resources

- Model repositories
  - https://huggingface.co/models
  - https://tfhub.dev/
  - https://huggingface.co/ckiplab
  - https://huggingface.co/ckip-joint
- Tutorial on transfer learning
  - https://www.tensorflow.org/tutorials/images/transfer_learning
  - https://github.com/jindongwang/transferlearning/tree/master/code/DeepDA
- Self-supervised learning
  - https://github.com/tensorflow/similarity
  - https://github.com/lightly-ai/lightly
  - https://github.com/KeremTurgutlu/self_supervised/tree/main
  - https://github.com/nlp-with-transformers/notebooks/tree/main
  - https://huggingface.co/docs/transformers/tasks/zero_shot_image_classification

Resources

- Foundation model
  - https://snorkel.ai/foundation-models/
  - https://github.com/facebookresearch/dinov2
  - https://github.com/facebookresearch/ImageBind
- Prompt engineering
  - https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/
  - https://blog.o-w-o.cc/archives/chatgpt-prompt-guideline
- Tutorials on other related topics
  - https://d2l.ai/chapter_natural-language-processing-pretraining/index.html
  - Multi-Task Learning
  - Zero-Shot Learning
  - A Survey on Contrastive Self-supervised Learning
  - Curriculum Learning

Resources

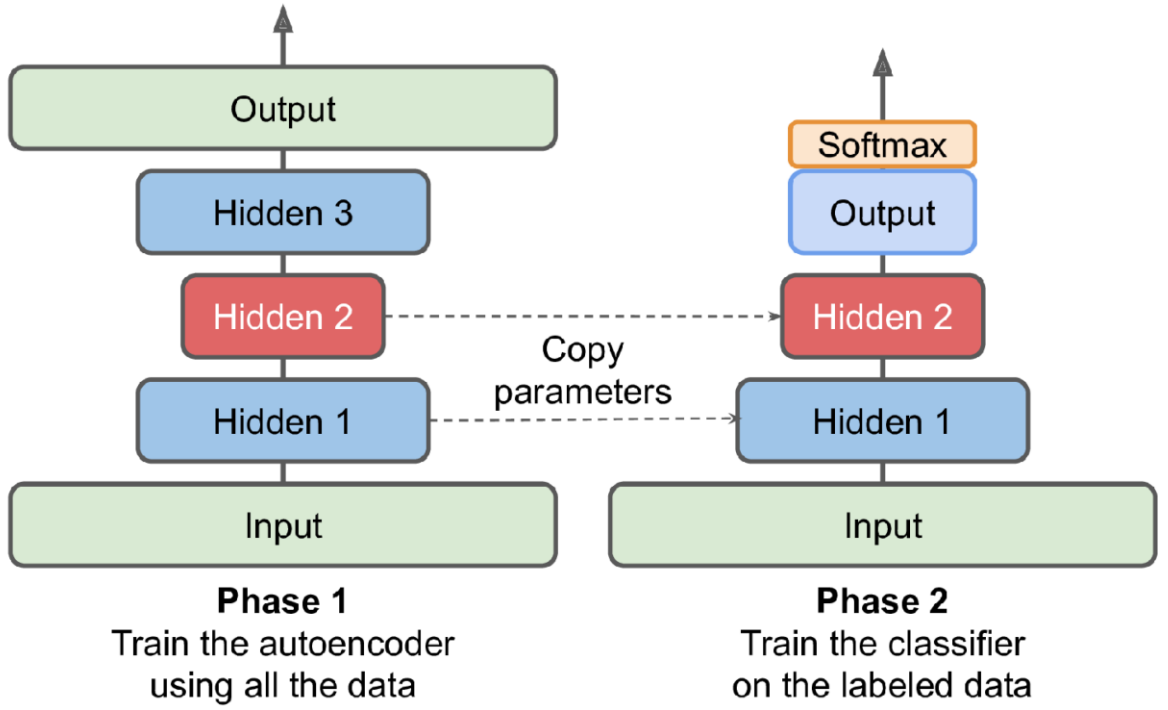
- Instruction tuning and Chain of Thought
  - https://www.youtube.com/watch?v=aZ_jXZvxyVg

Unsupervised pretraining

- In case you want to tackle a complex task for which you don't have much labeled data, but unfortunately you can't find a model trained on a similar task
  - If you can gather plenty of unlabeled training data, you can try to use it to train an unsupervised model, such as an autoencoder or a generative adversarial network
  - You can then reuse the lower layers of the autoencoder or GAN's discriminator, add the output layer for your task on top, and finetune the final network with the labeled training examples

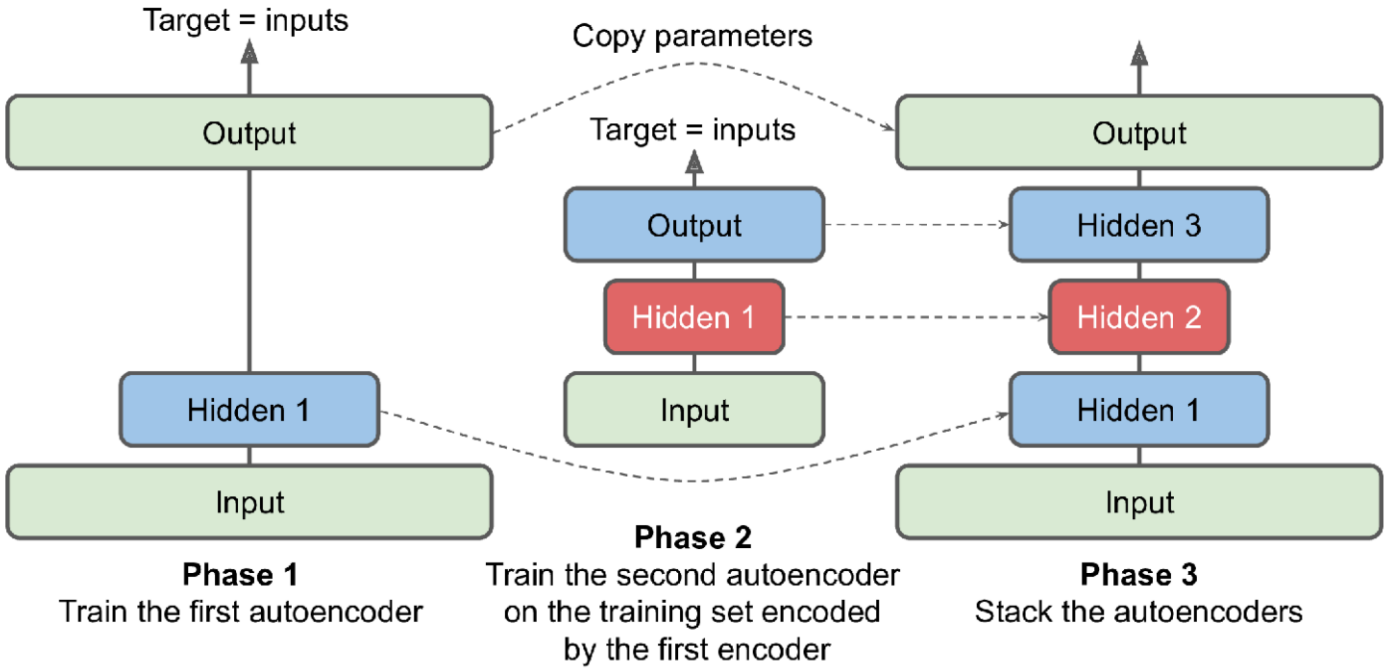
Unsupervised pretrained

- If you have a large dataset but most of it is unlabeled, you can first train a stacked autoencoder using all the data
  - Then reuse the lower layers to create a neural network for your actual task and train it using the labeled data, you may also want to freeze the pretrained layers
  - When an autoencoder is neatly symmetrical, a common technique is to tie the weights of the decoder layers to the weights of the encoder layers. This halves the number of weights in the model, speeding up training and limiting the risk of overfitting



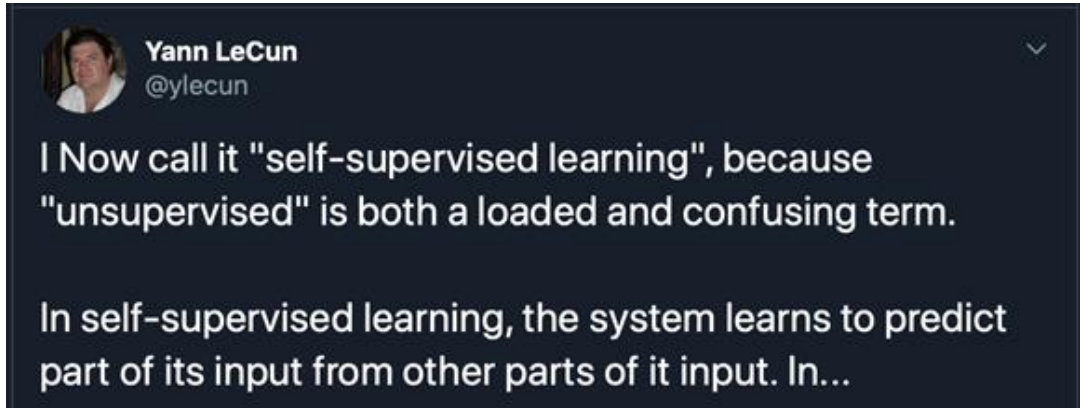
Training one autoencoder at a time

- It is possible to train one shallow autoencoder at a time, then stack all of them into a single stacked autoencoder called “greedy layerwise training”
  - During the first phase of training, the first autoencoder learns to reconstruct the inputs. Then we encode the whole training set using this first autoencoder, and this gives us a new (compressed) training set. We then train a second autoencoder on this new dataset. This is the second phase of training
  - Finally, we build a big sandwich using all these autoencoders, This gives us the final stacked autoencoder



What is Self-Supervised Learning?

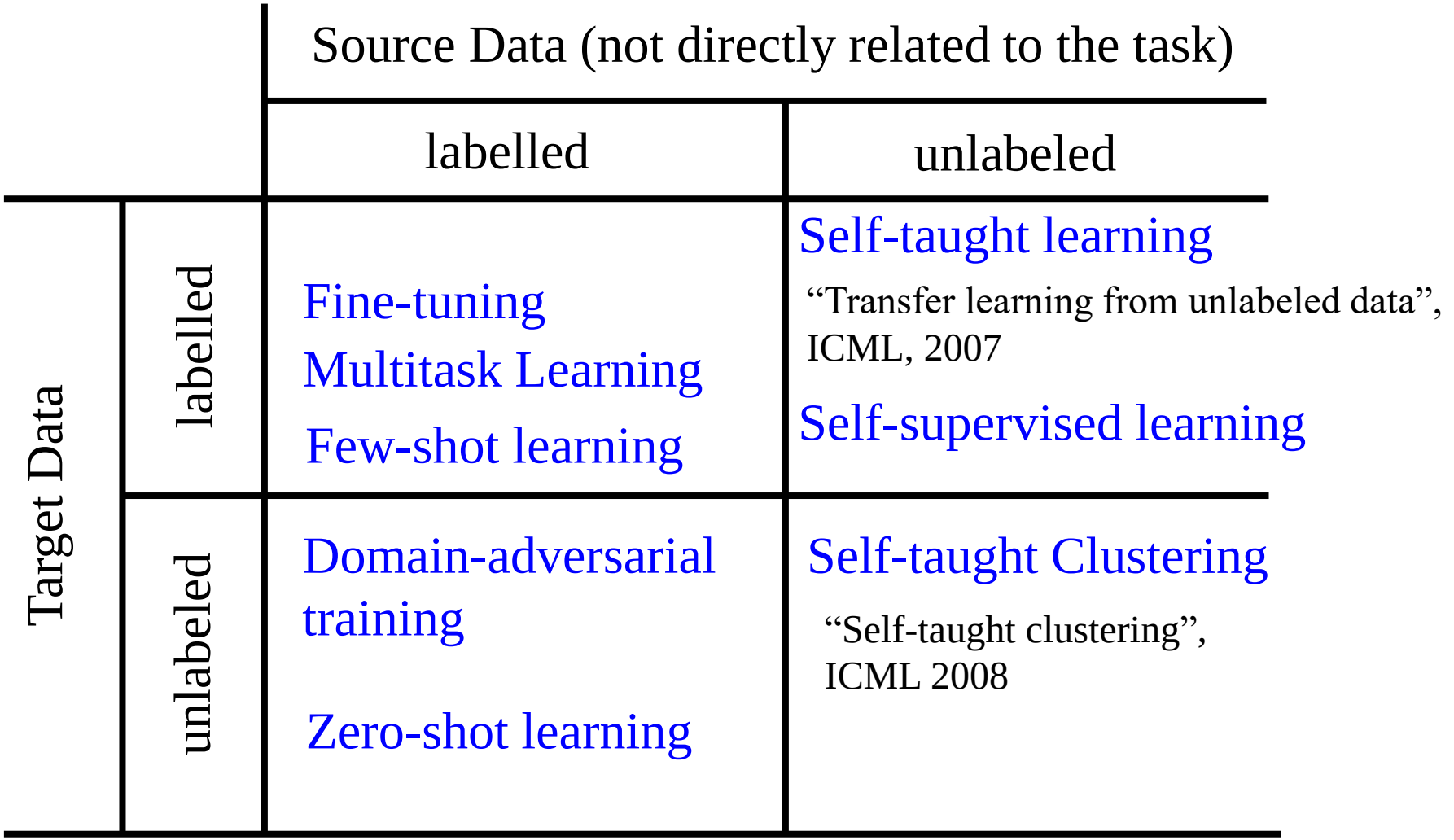
- A version of unsupervised learning where data provides the supervision.



- In general, withhold some part of the data and the task a neural network to predict it from the remaining parts.

- Goal: Learning to represent the world before learning tasks.

Taxonomy of Transfer learning



https://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/transfer.pptx

Domain adaptation

Training Data (Source domain)



Testing Data





Target domain

99.5%

57.5%

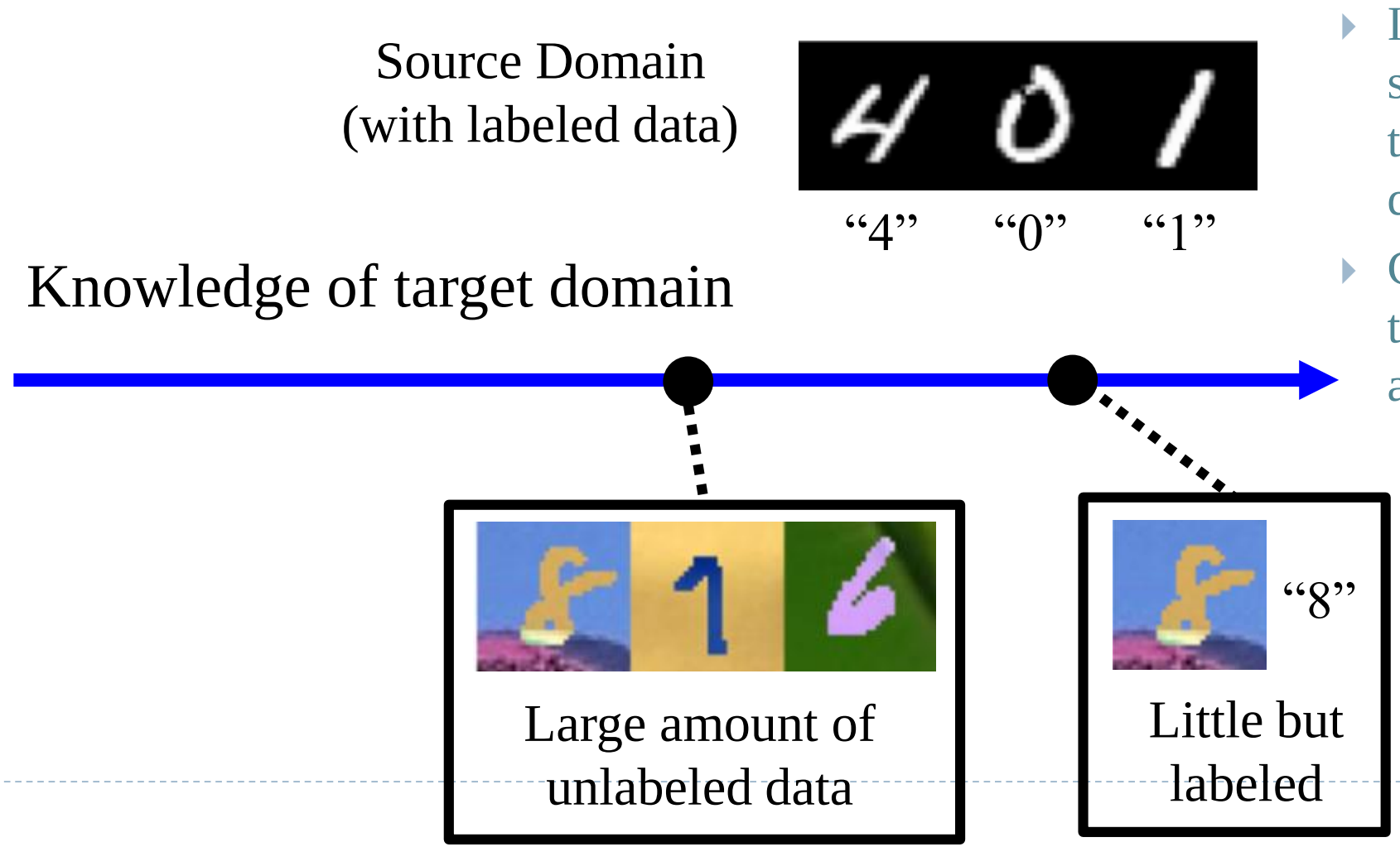
The results are from: http://proceedings.mlr.press/v37/ganin15.pdf

Domain shift: Training and testing data have different distributions.

→ **Domain adaptation**

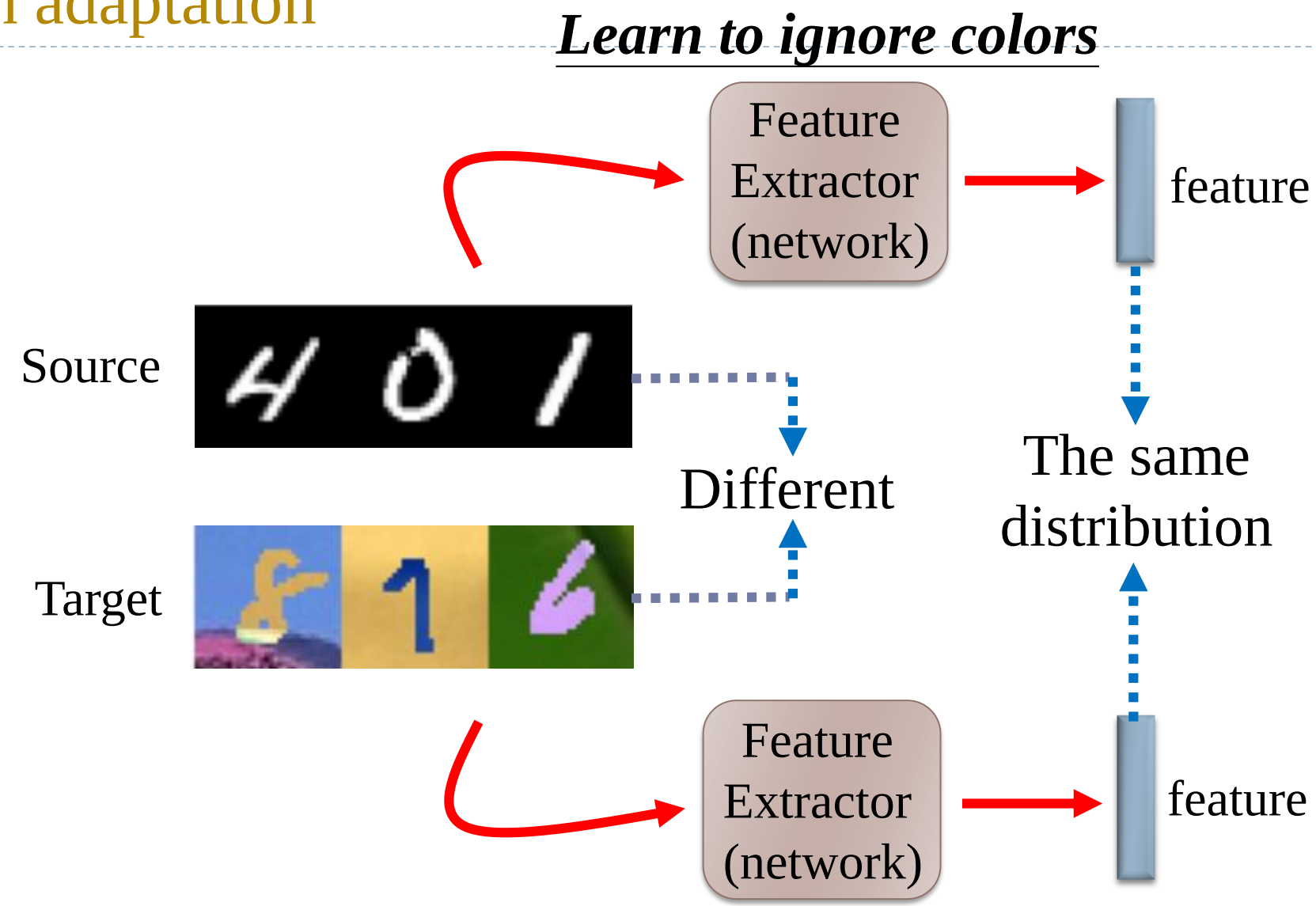
Domain adaptation

- https://paperswithcode.com/sota/domain-adaptation-on-gta5-to-cityscapes

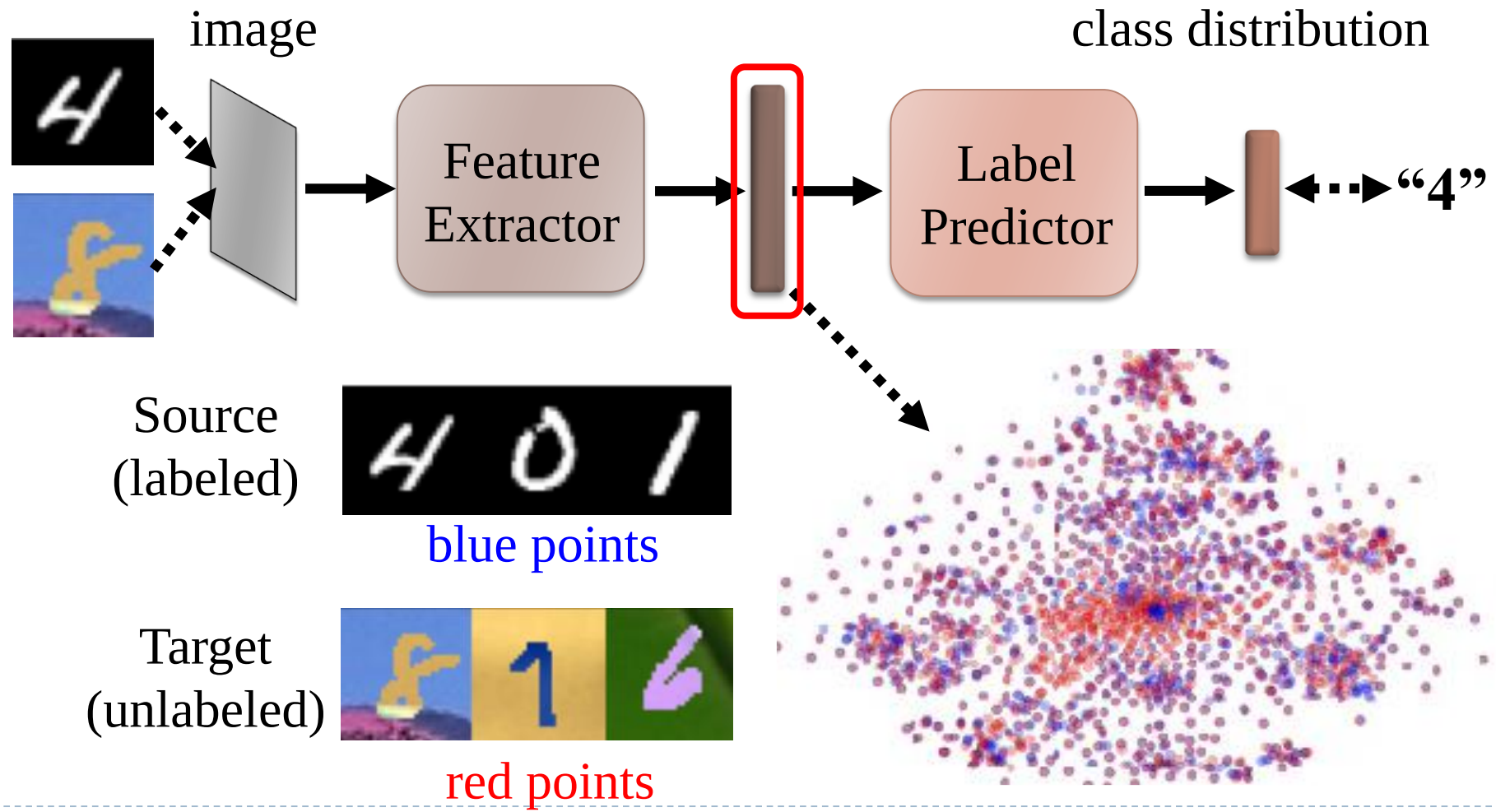


- Idea: training a model by source data, then fine-tune the model by target data
- Challenge: only limited target data, so be careful about overfitting

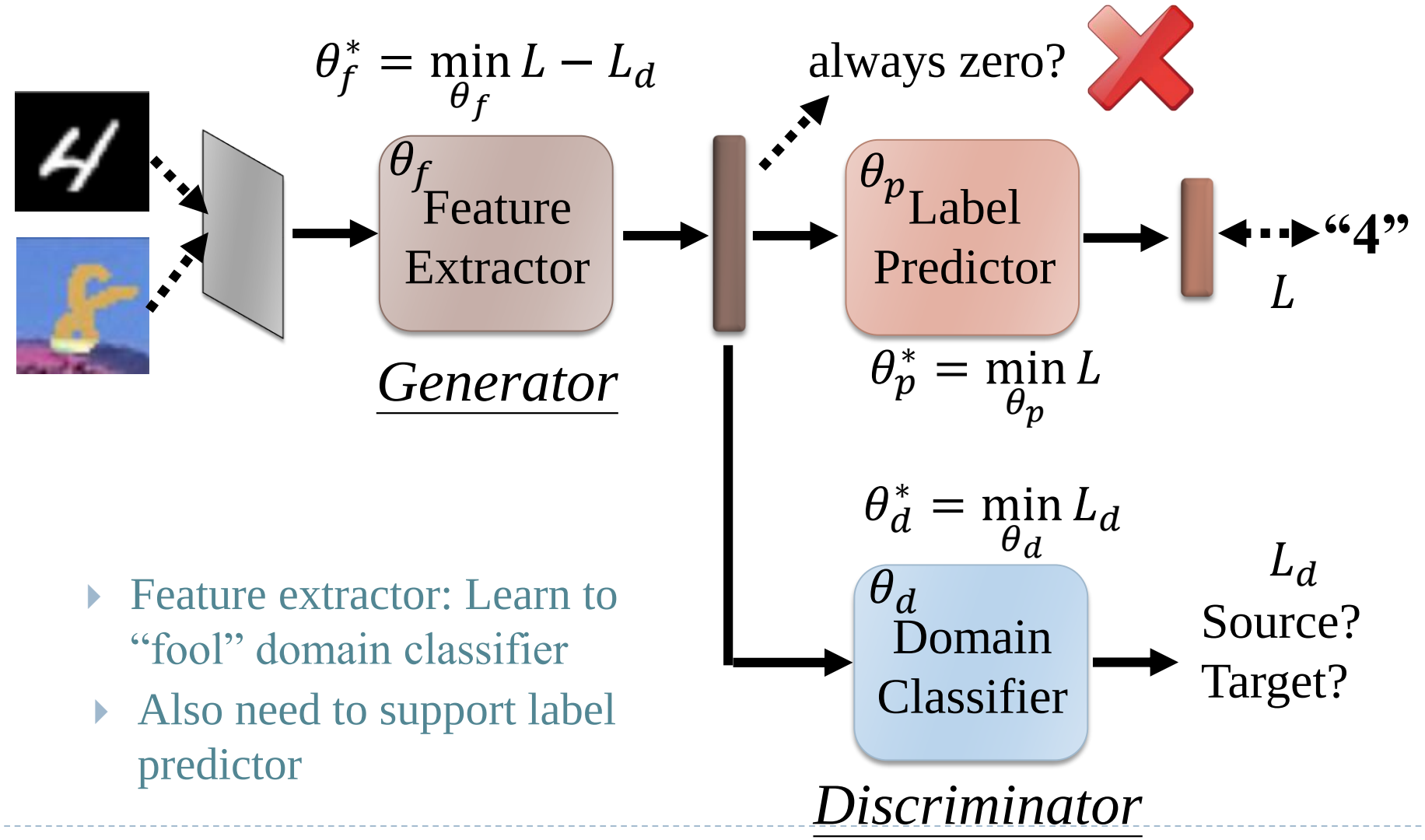
Domain adaptation



Domain adversarial training



Domain adversarial training



- Feature extractor: Learn to "fool" domain classifier
- Also need to support label predictor

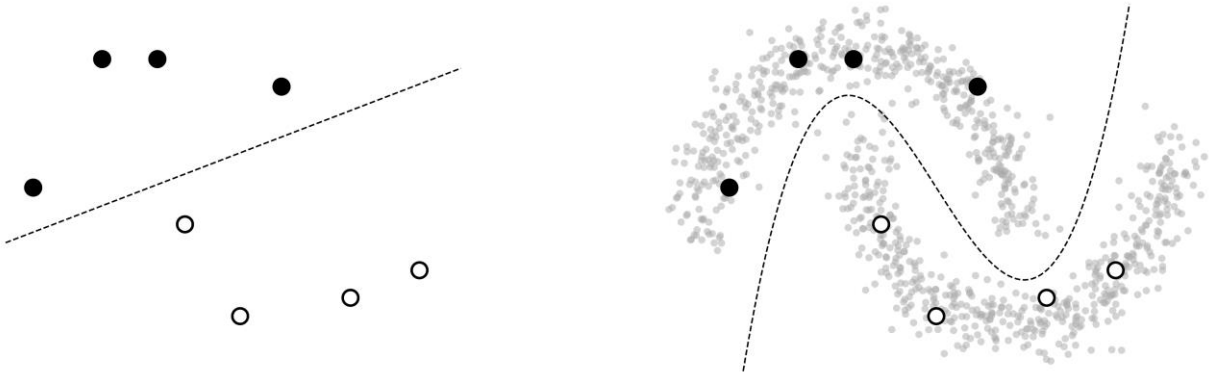
Domain adversarial training



METHOD	SOURCE / TARGET	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY		.5749	.8665	.5919	.7400
PROPOSED APPROACH		**.8149** (57.9%)	**.9048** (66.1%)	**.7107** (29.3%)	**.8866** (56.7%)
TRAIN ON TARGET		.9891	.9244	.9951	.9987

Semi-supervised learning

- Semi-supervised learning can alleviate the need for labeled data by taking advantage of unlabeled data
  - The general goal of semi-supervised learning is to allow the model to learn the high-level structure of the data distribution from unlabeled data and only rely on the labeled data for learning the fine-grained details of a given task
  - Whereas in standard supervised learning we assume that we have access to samples from the joint distribution of data and labels $x, y \sim p(x, y)$, semi-supervised learning assumes that we additionally have access to samples from the marginal distribution of $x \sim p(x)$



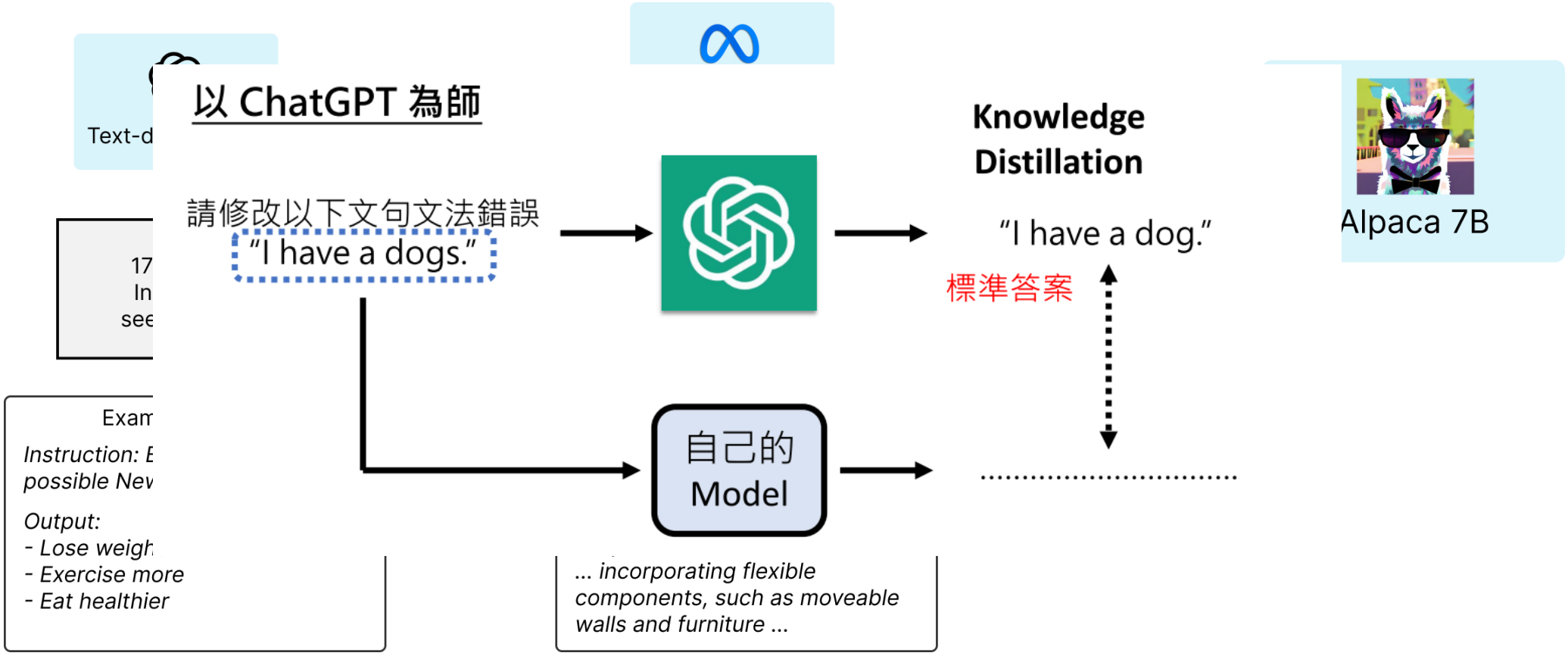
Semi-supervised learning: self-training and pseudo-labeling

- A straightforward approach to *semi-supervised learning* is self-training
  - The basic idea behind self-training is to use the model itself to infer predictions on unlabeled data, and then treat these predictions as labels for subsequent training
  - Recently, it has become common to refer to this approach as "pseudo-labeling" because the inferred labels for unlabeled data are only "pseudo-correct" in comparison with the true, ground-truth targets used in supervised learning
  - A common strategy is to use a "selection metric" which tries to only retain pseudo-labels that are correct. For example, assuming that a model outputs probabilities for each possible class, a frequently-used selection metric is to only retain pseudo-labels whose largest class probability is above a threshold
  - Also refer to noisy student approach, some recent paper advocate self-training approach rather than supervised or self-supervised way, see here

Build your own ChaptGPT

Empirically derived formulas for optimal model and training set size given a fixed compute budget (Open-source, but non-commercial)

- Alpaca



https://github.com/FreedomIntelligence/LLMZoo

Prompt Engineering



重要程度 Importance

* **Task** 工作指令

Context 背景資訊

Exemplar 範例資訊

Persona 角色扮演

Format 輸出格式

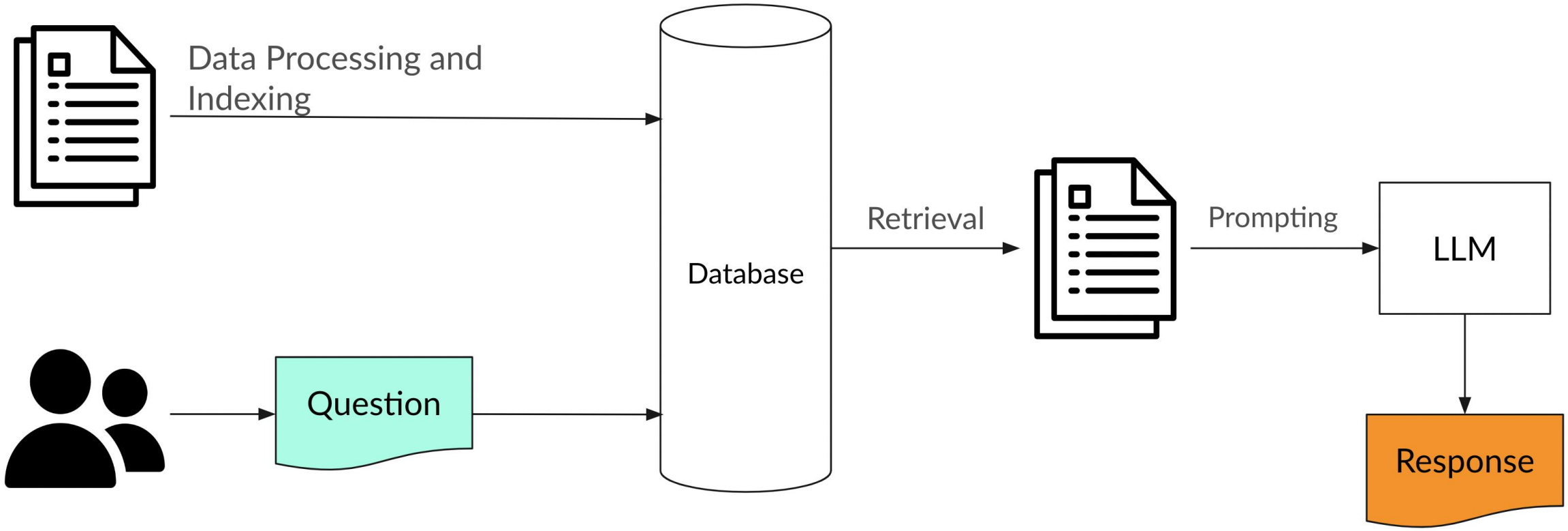
Tone 語氣風格



高 High

低 Low

Retrieval Augmented Generation (RAG)



Multi-agent Task Solver

