torchdistill Meets Hugging Face Libraries for Reproducible, Coding-Free Deep Learning Studies: A Case Study on NLP

Yoshitomo Matsubara *
University of California, Irvine
yoshitom@uci.edu

Abstract

Reproducibility in scientific work has been becoming increasingly important in research communities such as machine learning, natural language processing, and computer vision communities due to the rapid development of the research domains supported by recent advances in deep learning. In this work, we present a significantly upgraded version of torchdistill[1], a modular-driven coding-free deep learning framework significantly upgraded from the initial release, which supports only image classification and object detection tasks for reproducible knowledge distillation experiments. To demonstrate that the upgraded framework can support more tasks with third-party libraries, we reproduce the GLUE benchmark results of BERT models using a script based on the upgraded torchdistill, harmonizing with various Hugging Face libraries. All the 27 fine-tuned BERT models and configurations to reproduce the results are published at Hugging Face[2], and the model weights have already been widely used in research communities. We also reimplement popular small-sized models and new knowledge distillation methods and perform additional experiments for computer vision tasks.

1 Introduction

The rapid developments of various research domains such as natural language procession (NLP), computer vision, and speech recognition (He et al., 2016; Ballé et al., 2017; Devlin et al., 2019; Dosovitskiy et al., 2020; Raffel et al., 2020; Rombach et al., 2022; Radford et al., 2023) have been supported by advances in deep learning (Krizhevsky et al., 2012; Mikolov et al., 2013; Kingma and Welling, 2014; Sutskever et al., 2014; Kingma and Ba, 2015; Sohl-Dickstein et al., 2015; Vaswani et al., 2017; Brown et al., 2020). While it has been developed rapidly, poor reproducibility of deep learning-based studies is a severe problem that research communities have been facing (Crane, 2018; Yang et al., 2019; Daoudi et al., 2021; Matsubara, 2021), and the reproducibility has been attracting significant attention from researchers (Gundersen et al., 2018; Gundersen, 2019; Dodge et al., 2019; Kamphuis et al., 2020; Lopresti and Nagy, 2021; Pineau et al., 2021).

To address the serious problem, research communities introduced reproducibility checklists. At the time of writing, some venues require authors to complete checklists when submitting their work *e.g.*, Responsible NLP Research Checklist[3] (Rogers et al., 2021) at NLP venues (ACL, NAACL, ARR) and Paper Checklist at NeurIPS.[4]

Matsubara (2021) developed torchdistill, a modular, configuration-driven knowledge distillation framework built on PyTorch (Paszke et al., 2019) for reproducible deep learning research. Knowledge distillation (Hinton et al., 2014) is a well known model compression method usually to train a small model (called *student*) leveraging outputs from a more complex model (called *teacher*) as part of loss functions to be minimized. Recent knowledge distillation approaches are more complex *e.g.*, using intermediate layers' outputs (embeddings or feature maps) besides the final output (logits) of teacher models with auxiliary module branches attached to teacher and/or student models during training (Kim et al., 2018; Zhang et al., 2020; Chen et al., 2021), using multiple teachers (Mirzadeh et al., 2020; Matsubara et al., 2022b), and training multilingual or non-English models solely with an English teacher model (Reimers and Gurevych, 2020; Li et al., 2022b; Gupta et al., 2023).

For implementing such approaches, researchers

*This work was done prior to joining Amazon.

[1]`https://github.com/yoshitomo-matsubara/torchdistill/`

[2]`https://huggingface.co/yoshitomo-matsubara`

[3]`https://aclrollingreview.org/responsibleNLPresearch/`

[4]`https://neurips.cc/public/guides/PaperChecklist`

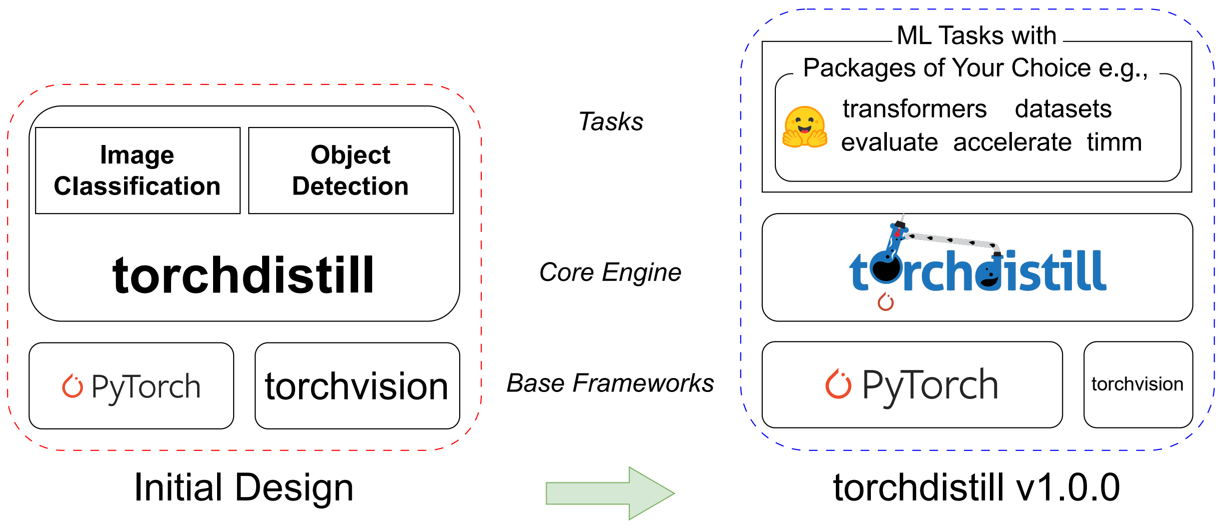

Figure 1: Initial design of torchdistill (Matsubara, 2021) vs. v1.0.0 in this work.

unpacked existing model implementations and modified their input-output interfaces to extract and/or hard-code new auxiliary modules (trainable modules to be used only during training) (Zagoruyko and Komodakis, 2016; Passalis and Tefas, 2018; Heo et al., 2019; Park et al., 2019; Tian et al., 2019; Xu et al., 2020; Chen et al., 2021). torchdistill (Matsubara, 2021) was initially designed as a unified knowledge distillation framework to enable users to design experiments by declarative PyYAML configuration files without such hardcoding effort and help researchers complete the ML Code Completeness Checklist[5] for high-quality reproducible knowledge distillation studies. One of its key concepts is that a declarative PyYAML configuration file designs an experiment and explains key hyperparameters and components used in the experiment. While the initial framework is well generalized and supports 18 different knowledge distillation methods implemented in a unified way, the implementation of the initial framework is highly dependent on torchvision[6], a package for popular datasets, model architectures, and common image transformations for computer vision tasks.

In this work, we significantly upgrade torchdistill from the initial framework (Matsubara, 2021) to enable further generalized implementations, supporting more flexible module abstractions and enhance the advantage of decralative PyYAML configuration files to design experiments with third-party packages of user's choice, as promised in (Matsubara, 2021). Using GLUE tasks (Wang et al., 2019) as an example, we demonstrate that the upgraded torchdistill and a new script harmonize with Hugging Face Transformers (Wolf et al., 2020), Datasets (Lhoest et al., 2021), Accelerate (Gugger et al., 2022), and Evaluate (Von Werra et al., 2022) to reproduce the GLUE test results reported in (Devlin et al., 2019) by fine-tuning pretrained BERT-Base and BERT-Large models with the upgraded torchdistill. We also conduct knowledge distillation experiments using the fine-tuned BERT-Large models as teachers to train BERT-Base models. All these experiments are performed on Google Colaboratory.[7] We also publish all the code and configuration files at GitHub[1] and trained model weights and training logs at Hugging Face[2] for reproducibility and helping researchers build on this work. Our BERT models fine-tuned for the GLUE tasks have already been downloaded 138,000 times in total and widely used in research communities not only in research papers but also in tutorials of deep learning frameworks and ACL 2022. Besides the NLP tasks, we reimplement popular small-sized computer vision models and a few more recent knowledge distillation methods as part of torchdistill, and perform additional experiments to demonstrate that the upgraded torchdistill still supports computer vision tasks.

2 Related Work

In this section, we briefly summarize related work on open source software that supports end-to-end

[5]https://github.com/paperswithcode/releasing-research-code

[6]https://github.com/pytorch/vision

[7]https://colab.google/

research frameworks. Yang et al. (2018) propose Anserini, an information retrieval toolkit built on Lucene[8] for reproducible information retrieval research. Pyserini (Lin et al., 2021) is a Python toolkit built on PyTorch (Paszke et al., 2019) and Faiss (Johnson et al., 2019) for reproducible information retrieval research with sparse and dense representations, and the sparse representation-based retrieval support comes from Lucene via Anserini.

AllenNLP (Gardner et al., 2018) is a toolkit built on PyTorch for research on deep learning methods in NLP and designed to lower barriers to high quality NLP research *e.g.*, useful NLP module abstractions and defining experiments using declarative configuration files. Highly inspired by AllenNLP, Matsubara (2021) design torchdistill, a module, configuration-driven framework built on PyTorch for reproducible knowledge distillation studies. Similar to AllenNLP, torchdistill enables users to design experiments by declarative PyYAML configuration files and supports high-level module abstractions. For image classification and object detection tasks, its generalized starter scripts and configurations help users implement knowledge distillation methods without much coding cost. Matsubara (2021) also reimplement 18 knowledge distillation methods with torchdistill and point out that the standard knowledge distillation (Hinton et al., 2014) can outperform many of the recent state of the art knowledge distillation methods for a popular teacher-student pair (ResNet-34 and ResNet-18) with ILSVRC 2012 dataset (Russakovsky et al., 2015). In Section 3, we describe major upgrades in torchdistill from the initial release (Matsubara, 2021).

3 Major Upgrades from the Initial Release

In this section, we summarize the major upgrades from the initial release of torchdistill (Matsubara, 2021). Figure 1 highlights high-level differences between the initial design (Matsubara, 2021) of torchdistill and a largely upgraded version in this work. The initial torchdistill is dependent on PyTorch and torchvision and contains key modules and functionalities specifically designed to support image classification and object detection tasks. For example, dataset modules that the initial version officially supports are only those in torchvision, and some of dataset-relevant functionalities such as building a sequence of data transforms and dataset loader are based on datasets in torchvision.

In this work, we make torchdistill less dependent on torchvision and support more tasks with third-party packages of users' choice, by generalizing some of the key components in the framework and exporting task-specific implementations to the corresponding executable scripts and local packages. We also reimplement popular small-sized models whose official PyTorch implementations are not either available or maintained.

3.1 PyYAML-based Instantiation

A declarative PyYAML configuration file plays an important role in torchdistill. Users can design experiments with the declarative PyYAML configuration file, which defines various types of abstracted modules with hyperparameters such as dataset, model, optimizer, scheduler, and loss modules. To allow more flexibility in PyYAML configurations, we add more useful constructors such as importing arbitrary local packages to register modules but without edits on an executable script, and instantiating an arbitrary class with a log message. Those can be done simply at the very beginning of an experiment when loading the PyYAML configuration file and make the configuration files more self-explanatory since the configuration format used for the initial version does not explicitly tell users whether the experiment needs specific local packages. Those features also help us generalize ways to define key module such as datasets and their components (*e.g.*, pre-processing transforms, samplers).

Figure 2 shows an example that build a sequence of image/tensor transforms with the initial version and torchdistill in this work. While the former requires both a Python function specifically designed for torchvision modules (`build_transform`) and a list of dict objects defined in a PyYAML configuration to be given to the function as (`transform_params_config`), the latter can build exactly the same transform when loading the PyYAML configuration and store the instantiated object as part of a dict object with **`transform`** key.

3.2 Generalized Modules for Supporting More Tasks

The PyYAML-based instantiation feature described in Section 3.1 enables us to remove torchvision-specific modules mentioned in Section 3 (*e.g.*, `build_transform` in Fig. 2) so that we can reduce

[8] https://lucene.apache.org/

```python
import torchvision
from torchdistill.datasets.transform import TRANSFORM_CLASS_DICT

TRANSFORM_CLASS_DICT.update(torchvision.transforms.__dict__)


def build_transform(transform_params_config, compose_cls=None):
    if not isinstance(transform_params_config, (dict, list)) or len(transform_params_config) == 0:
        return None

    component_list = list()
    if isinstance(transform_params_config, dict):
        for component_key in sorted(transform_params_config.keys()):
            component_config = transform_params_config[component_key]
            params_config = component_config.get('params', dict())
            if params_config is None:
                params_config = dict()

            component = TRANSFORM_CLASS_DICT[component_config['type']](**params_config)
            component_list.append(component)
    else:
        for component_config in transform_params_config:
            params_config = component_config.get('params', dict())
            if params_config is None:
                params_config = dict()

            component = TRANSFORM_CLASS_DICT[component_config['type']](**params_config)
            component_list.append(component)
    return transforms.Compose(component_list) if compose_cls is None else compose_cls(component_list)
```

```yaml
transform_params:
  - type: 'RandomCrop'
    params:
      size: 32
      padding: 4
  - type: 'RandomHorizontalFlip'
    params:
      p: 0.5
  - type: 'ToTensor'
    params:
  - type: 'Normalize'
    params:
      mean: [0.49139968, 0.48215841, 0.44653091]
      std: [0.24703223, 0.24348513, 0.26158784]
```

```yaml
transform: !import_call
  key: 'torchvision.transforms.Compose'
  init:
    kwargs:
      transforms:
        - !import_call
          key: 'torchvision.transforms.RandomCrop'
          init:
            kwargs:
              size: 32
              padding: 4
        - !import_call
          key: 'torchvision.transforms.RandomHorizontalFlip'
          init:
            kwargs:
              p: 0.5
        - !import_call
          key: 'torchvision.transforms.ToTensor'
          init:
        - !import_call
          key: 'torchvision.transforms.Normalize'
          init:
            kwargs:
              mean: [0.49139968, 0.48215841, 0.44653091]
              std: [0.24703223, 0.24348513, 0.26158784]
```

Figure 2: Example of two different ways to build a sequence of transforms in torchvision (**transform**) for CIFAR-10 dataset. The initial version (top, left) defines a function for torchvision `build_transform` in torchdistill and gives the function a list of dict objects in the left PyYAML as `transform_params_config`. torchdistill in this work (right) can build exactly the same **transform** by instantiating each of the transform classes step-by-step with `!import_call`, one of our pre-defined PyYAML constructors in the upgraded torchdistill.

torchdistill's dependency on torchvision and generalize its modules for supporting more tasks.

The initial version of torchdistill is designed to support image classification and object detection tasks based on torchvision, and torchvision models for the tasks such as ResNet (He et al., 2016) and Faster R-CNN (Ren et al., 2015) require an image (tensor) and an annotation as part of the model inputs during training. However, this interface does not generalize well to support other tasks. Taking a text classification task as an example, Transformer (Vaswani et al., 2017) models in Hugging Face Transformers (Wolf et al., 2020) have much more input data fields such as (not limited to) token IDs, attention mask, token type IDs, position IDs, and labels for BERT (Devlin et al., 2019), and different models have different input data fields *e.g.*, BART (Lewis et al., 2020) has additional input data fields such as token IDs for its decoder.

In order to support diverse models and tasks, we generalize interfaces of model input/output and the subsequent processes in torchdistill such as computing training losses. For demonstrating that the upgraded torchdistill can support more tasks, we provide starter scripts based on the upgraded framework for GLUE (Wang et al., 2019) and semantic segmentation tasks. For the GLUE tasks, we harmonize popular Python libraries with torchdistill in the script such as Hugging Face Transformers (Wolf et al., 2020), Datasets (Lhoest et al., 2021), and Evaluate (Von Werra et al., 2022) for model, dataset, and evaluation modules. We also leverage Accelerate (Gugger et al., 2022) for efficient training and inference. In Section 4.1, we demonstrate GLUE experiments with torchdistill and the third-party libraries.

3.3 Reimplemented Models and Methods

We find in recent knowledge distillation studies (Tian et al., 2019; Xu et al., 2020; Chen et al., 2021) that there is still a demand of small models for relatively simple datasets such as ResNet (He et al., 2016)[9], WRN (Zagoruyko and Komodakis, 2016)[10], and DenseNet (Huang et al., 2017)[11] for image classification tasks with CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009) since the official repositories are no longer maintained and/or not implemented with PyTorch.

For helping the community conduct better benchmarking, we reimplement the models for CIFAR-10 and CIFAR-100 datasets as part of torchdistill and attempt to reproduce the reported results following the original training recipes (See Section 4). With the upgraded torchdistill, we also reimplement and test a few more knowledge distillation methods (He et al., 2019; Chen et al., 2021).

4 Google Colab Demos

In this section, we demonstrate that the upgraded torchdistill can collaborate with third-party libraries for supporting more tasks. We also attempt to reproduce the CIFAR-10 and CIFAR-100 results reported in the original papers. To lower the barrier to reusing and building on the scripts with torchdistill, we conduct all the experiments on Google Colaboratory[7], which gives users access to GPUs free of charge. We publish the Jupyter Notebook[12] files to run the experiments as part of torchdistill repository[1] so that researchers can easily use them.

4.1 GLUE Tasks

The GLUE benchmark (Wang et al., 2019) uses nine datasets in three different task categories. The benchmark consists of 1) two single-sentence tasks: CoLA (Warstadt et al., 2019) and SST-2 (Socher et al., 2013), 2) three similarity and paraphrase tasks: MRPC (Dolan and Brockett, 2005), QQP[13], and STS-B (Cer et al., 2017), and 3) four inference tasks: MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016; Wang et al., 2019), RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al.), and WNLI (Levesque et al., 2012).

We attempt to reproduce GLUE test results reported in a popular study, BERT (Devlin et al., 2019), using the upgraded torchdistill harmonizing with Hugging Face libraries (transformers, datasets, evaluate, and accelerate) (Wolf et al., 2020; Lhoest et al., 2021; Von Werra et al., 2022; Gugger et al., 2022). Following the experiments, we also conduct knowledge distillation experiments that fine-tune pretrained BERT-Base models for GLUE tasks, using the fine-tuned BERT-Large models as teachers for the knowledge distillation method of Hinton et al. (2014) minimizing

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{CE}}(\hat{\mathbf{y}}, \mathbf{y}) + (1-\alpha) \cdot \tau^2 \cdot \mathcal{L}_{\text{KL}}\left(\mathbf{p}, \mathbf{q}\right), \quad (1)$$

where $\mathcal{L}_{\text{CE}}$ is a standard cross entropy. $\hat{\mathbf{y}}$ indicates the student model's estimated class probabilities, and $\mathbf{y}$ is the annotated category. $\mathcal{L}_{\text{KL}}$ is the Kullback-Leibler divergence, and α and τ are a balancing factor and a temperature, respectively. $\mathbf{p}$ and $\mathbf{q}$ represent the *softened* output distributions from teacher and student models, respectively. $\mathbf{p}$ is used as a target distribution for $\mathcal{L}_{\text{KL}}$. Specifically, $\mathbf{p} = [p_1, p_2, \ldots, p_{|\mathcal{C}|}]$ where $\mathcal{C}$ is a set of categories in the target task. p_i indicates the student model's softened output value (scalar) for the i-th category:

$$p_i = \frac{\exp\left(\frac{v_i}{\tau}\right)}{\sum_{k \in \mathcal{C}} \exp\left(\frac{v_k}{\tau}\right)}, \quad (2)$$

[9]https://github.com/facebookarchive/fb.resnet.torch

[10]https://github.com/szagoruyko/wide-residual-networks

[11]https://github.com/liuzhuang13/DenseNet

[12]https://jupyter.org/

[13]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

Model (Method, Reference)	MNLI-(m/mm) Acc./Acc.	QQP F1	QNLI Acc.	SST-2 Acc.	CoLA M Corr.	STS-B P-S Corr.	MRPC F1	RTE Acc.	WNLI Acc.
BERT-Large (FT, Devlin et al. (2019))	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	N/A
BERT-Large (FT, Ours)	86.4/85.7	72.2	92.4	94.6	61.5	85.0	89.2	68.9	65.1
BERT-Base (FT, Devlin et al. (2019))	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	N/A
BERT-Base (FT, Ours)	84.2/83.3	71.4	91.0	94.1	51.1	84.4	86.8	66.7	65.8
BERT-Base (KD, Ours)	85.9/84.7	72.8	90.7	93.7	57.0	85.6	87.5	66.7	65.1

Table 1: GLUE test results. Our results are hyperlinked to our Hugging Face Model repositories. FT: Fine-Tuning, KD: Knowledge Distillation using BERT-Large (FT, ours) as teacher.

where τ is one of the hyperparameters defined in Eq. (1). v_i denotes a logit value for the i-th category. The same rules are applied to $\mathbf{q}$ for the student model.

For reproducing the GLUE test results in (Devlin et al., 2019), we use pretrained BERT-Base[14] and BERT-Large[15] models in Hugging Face Transformers (Wolf et al., 2020). Following (Devlin et al., 2019) we minimize a standard cross-entropy and the Adam optimizer (Kingma and Ba, 2015) with slightly extended hyperparameter choices: batch size of either 16 or 32 and 2-5 epochs for fine-tuning and select a learning rate among $\{2.0\times10^{-5}, 3.0\times10^{-5}, 4.0\times10^{-5}, 5.0\times10^{-5}\}$ on the *dev* set for each of the tasks. For knowledge distillation, we also choose learning rate from $\{1.0 \times 10^{-5}, 2.0 \times 10^{-5}, 3.0 \times 10^{-5}, 4.0 \times 10^{-5}, 5.0 \times 10^{-5}\}$, temperature $\tau \in \{1, 3, 5, 7, 9, 11\}$, and a balancing weight $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ based on the *dev* sets. Note that since STS-B is not a classification task, we use the sum of 1) a mean squared error between the annotation and the student model's output and 2) a mean squared error between outputs of the teacher and student models instead of Eq. (1) for the dataset.

Table 1 shows the GLUE test results reported by Devlin et al. (2019) and those obtained from GLUE Benchmark[16] for our three configurations: fine-tuning pretrained BERT-Base (FT, Ours) and pretrained BERT-Large (FT, Ours) models and knowledge distillation to fine-tune pretrained BERT-Base (KD, Ours) as a student, using the fine-tuned BERT-Large as the teacher. Note that Devlin et al. (2019) do not report the results for the WNLI test dataset.

Overall, our fine-tuned BERT-Base and BERT-Large models achieved GLUE test results comparable to the official test results reported by Devlin et al. (2019). The knowledge distillation method (Hinton et al., 2014) helped BERT-Base models improve the performance for most of the tasks, compared to those fine-tuned without the teacher models. All the trained model weights and training logs are published at Hugging Face[2], and the training configurations are published as part of the torchdistill GitHub repository.[1]

The fine-tuned BERT models we published are widely used in the research communities and have already been downloaded about 138,000 times in total at the time of writing. For instance, some of the models are used for benchmarks, ensembling, model quantization, token pruning (Matena and Raffel, 2022; Church et al., 2022; Guo et al., 2022; Lee et al., 2022), DeepSpeed Tutorials[17], Intel® Neural Compressor Examples[18], and ACL 2022 Tutorial.[19]

4.2 CIFAR-10 and CIFAR-100

We also attempt to reproduce the CIFAR-10 and CIFAR-100 results reported in (He et al., 2016; Zagoruyko and Komodakis, 2016; Huang et al., 2017) using the upgraded torchdistill with the reimplemented ResNet, WRN, and DenseNet models. We follow the original papers and reuse the hyperparameter choices and training recipes such as data augmentations. Note that we do not confider models that can not fit to the GPU memory which Google Colab can offer *e.g.*, ResNet-1202 (He et al., 2016) for CIFAR-10 and DenseNet-BC($k = 24$ and $k = 40$) (Huang et al., 2017) for CIFAR-10 and CIFAR-100.

Tables 2 and 3 compare the results reported in the original papers (He et al., 2016; Zagoruyko and Komodakis, 2016; Huang et al., 2017) with

[14]https://huggingface.co/bert-base-uncased
[15]https://huggingface.co/bert-large-uncased
[16]https://gluebenchmark.com/

[17]https://www.deepspeed.ai/tutorials/model-compression/
[18]https://github.com/intel/neural-compressor/tree/master/examples
[19]https://github.com/kwchurch/ACL2022_deepnets_tutorial

CIFAR-10 Model	Test Accuracy Original	Test Accuracy torchdistill
ResNet-20	91.25	91.92
ResNet-32	92.49	93.03
ResNet-44	92.83	93.20
ResNet-56	93.03	93.57
ResNet-110	93.57	93.50
WRN-40-4	95.47	95.24
WRN-28-10	96.00	95.53
WRN-16-8	95.73	94.76
DenseNet-BC (k=12, depth=100)	95.49	95.53

Table 2: CIFAR-10 results for ResNet (He et al., 2016), WRN (Zagoruyko and Komodakis, 2016), and DenseNet (Huang et al., 2017).

CIFAR-100 Model	Test Accuracy Original	Test Accuracy torchdistill
WRN-40-4	79.82	79.44
WRN-28-10	80.75	81.27
WRN-16-8	79.57	79.26
DenseNet-BC (k=12, depth=100)	77.73	77.14

Table 3: CIFAR-100 results for WRN (Zagoruyko and Komodakis, 2016) and DenseNet (Huang et al., 2017).

those we reproduced for CIFAR-10 and CIFAR-100 test datasets, respectively. We can confirm that for most of the reimplemented models, our results are comparable to those reported in the original papers. Those model weights and training configuration files are publicly available, and users can automatically download the weights via the upgraded torchdistill PyPI package.

5 ILSVRC 2012

As highlighted in Section 3, torchdistill was initially focused on supporting implementations of diverse knowledge distillation in a unified way and dependent on torchvision to specifically support image classification and object detection tasks with its relevant modules (see Fig. 1). To demonstrate that the upgraded torchdistill still preserves the feature, we reimplement a few more knowledge distillation methods with the upgraded torchdistill: knowledge review (KR) framework (Chen et al., 2021) and knowledge translation and adaptation with affinity distillation (KTAAD) (He et al., 2019). Note that Matsubara (2021) present the results of various knowledge distillation methods reimplemented with the initial version of torchdistill for ILSVRC 2012 and COCO 2017 (Lin et al., 2014) datasets. Those results are not included in this work, and we refer interested readers to (Matsubara, 2021).

T: ResNet-34 CE	S: ResNet-18 CE	S: ResNet-18 KR (Original)	S: ResNet-18 KR (Ours)
73.31	69.75	71.61	71.64

Table 4: ILSVRC 2012 top-1 accuracy of ResNet-18 (student) trained by KR (Chen et al., 2021) with pre-trained ResNet-34 (teacher). CE: torchvision models pretrained with cross-entropy.

Chen et al. (2021) demonstrate that the KR method can outperform other knowledge distillation using ResNet-34 and ResNet-18 (He et al., 2016), a popular pair of teacher and student models for the ImageNet (ILSVRC 2012) dataset (Russakovsky et al., 2015). Using the reimplemented KR method based on the upgraded torchdistill with hyperparameters in (Chen et al., 2021), we successfully reproduce their reported result of ResNet-18 for the ImageNet dataset as shown in Table 4. The trained model weights and configuration are published as part of the torchdistill repository.[1]

6 PASCAL VOC 2012 & COCO 2017

The initial torchdistill (Matsubara, 2021) supports image classification and object detection tasks. As mentioned in Section 3.2, we also provide a starter script for semantic segmentation tasks. Using two popular datasets, PASCAL VOC 2012 (Everingham et al., 2012) and COCO 2017 (Lin et al., 2014), we demonstrate that the upgraded torchdistill supports semantic segmentation tasks as well.

In the experiments with PASCAL VOC 2012 dataset, we use DeepLabv3 (Chen et al., 2017) with ResNet-50 and ResNet-101 backbones (He et al., 2016), using torchvision's pretrained model weights for COCO 2017 dataset. We choose hyperparameters such as learning rate policy and crop size based on the original study of DeepLabv3 (Chen et al., 2017). Our results are shown in Table 5, and DeepLabv3 with ResNet-101 achieved comparable mIoU (mean Intersection over Union) to the best DeepLabv3 model for PASCAL VOC 2012 dataset (*val* set) in the original study (mIoU: 82.70). Following torchvision documentation[20], we measure global pixelwise accuracy as well. In terms of both the metrics, DeepLabv3 with ResNet-101 outperforms DeepLabv3 with ResNet-50.

[20] https://pytorch.org/vision/stable/models.html#table-of-all-available-semantic-segmentation-weights

Model	mean IoU	Pixelwise Acc.
DeepLabv3 w/ ResNet-50	80.6	95.7
DeepLabv3 w/ ResNet-101	82.4	96.2

Table 5: PASCAL VOC 2012 (Segmentation, *val* set) validation results for DeepLabv3 with ResNet backbones (Chen et al., 2017) initialized with torchvision pretrained model weights for COCO 2017 dataset.

Method	mean IoU	Pixelwise Acc.
CE (torchvision)	57.9	91.2
KTAAD (Ours)	58.2	92.1

Table 6: COCO 2017 (Segmentation, *val* set) results for LRASPP with MobileNetV3-Large backbone (Howard et al., 2019).

We also examine our reimplemented KTAAD method (He et al., 2019) for the Lite R-ASPP model (LRASPP in torchvision) (Howard et al., 2019) as a student model, using the COCO 2017 dataset and the pretrained DeepLabv3 with ResNet-50 in torchvision as a teacher model, whose mIoU and global pixelwise accuracy are 66.4 and 92.4, respectively. Since the KTAAD method is not tested on COCO 2017 dataset for LRASPP with MobileNetV3-Large backbone in the original paper of KTAAD (He et al., 2019), our hyperparameter choice is based on torchvision's reference script.[21]

Table 6 presents the semantic segmentation results of LRASPP with MobileNetV3-Large backbone trained without the teacher model and by the KTAAD method we reimplemented. We confirm that the student model trained by KTAAD outperforms the same model trained on COCO 2017 available in torchvision in terms of mean IoU and global pixelwise accuracy.

As with other experiments, the trained model weights and configuration used in this section are published as part of the torchdistill repository.[1]

7 Conclusion

In this work, we significantly upgraded torchdistill (Matsubara, 2021), a modular, configuration-driven framework built on PyTorch (Paszke et al., 2019) for reproducible deep learning and knowledge distillation studies. We enhanced PyYAML-based instantiation, generalized internal modules for supporting more tasks, and reimplemented popular models and methods.

[21]https://github.com/pytorch/vision/tree/main/references/segmentation

To demonstrate that the upgraded framework can support more tasks as we claim, we provided starter scripts for new tasks based on the upgraded framework. One of the new starter scripts supports GLUE tasks (Wang et al., 2019) and harmonizes with Hugging Face Transformers (Wolf et al., 2020), Datasets (Lhoest et al., 2021), Accelerate (Gugger et al., 2022), and Evaluate (Von Werra et al., 2022). Using the script on Google Colaboratory, we reproduced the GLUE test results of fine-tuned BERT models (Devlin et al., 2019) and performed knowledge distillation experiments with our fine-tuned BERT-Large models as teacher models. Similarly, we reproduced CIFAR-10 and -100 results of popular small-sized models we reimplemented, using Google Colaboratory. Furthermore, we reproduced the result of ResNet-18 trained with the reimplemented KR method (Chen et al., 2021) for the ImageNet dataset. We also demonstrated a new starter script for semantic segmentation tasks using PASCAL VOC 2012 and COCO 2017 datasets, and the reimplemented KTAAD method (He et al., 2019) improves a pretrained semantic segmentation model in torchvision.

In this study, we also published 27 trained models for NLP tasks[2] and 14 trained models for computer vision tasks.[1] According to Hugging Face Model repositories, the BERT models fine-tuned for the GLUE tasks have already been downloaded about 138,000 times in total at the time of writing. Research communities leverage torchdistill not only for knowledge distillation studies (Liu et al., 2021; Li et al., 2022a; Lin et al., 2022; Dong et al., 2022; Miles and Mikolajczyk, 2023), but also for machine learning reproducibility challenge (MLRC) (Lee and Lee, 2023) and reproducible deep learning studies (Matsubara et al., 2022a,c; Furutanpey et al., 2023b,a; Matsubara et al., 2023). torchdistill is publicly available as a pip-installable PyPI package and will be maintained and upgraded for encouraging coding-free reproducible deep learning and knowledge distillation studies.

Acknowledgements

We thank the anonymous reviewers for their comments. This project has been supported by Travis CI's OSS credits and JetBrain's Free License Programs (Open Source) since November 2021 and June 2022, respectively.

References

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. 2017. End-to-end Optimized Image Compression. In *International Conference on Learning Representations*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The Sixth PASCAL Recognizing Textual Entailment Challenge.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. *Advances in neural information processing systems*, 33:1877–1901.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*.

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. 2021. Distilling Knowledge via Knowledge Review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5008–5017.

Kenneth Church, Valia Kordoni, Gary Marcus, Ernest Davis, Yanjun Ma, and Zeyu Chen. 2022. A Gentle Introduction to Deep Nets and Opportunities for the Future. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 1–6.

Matt Crane. 2018. Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

Nadia Daoudi, Kevin Allix, Tegawendé F Bissyandé, and Jacques Klein. 2021. Lessons Learnt on Reproducibility in Machine Learning Based Android Malware Detection. *Empirical Software Engineering*, 26(4):74.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show Your Work: Improved Reporting of Experimental Results, author=Dodge, Jesse and Gururangan, Suchin and Card, Dallas and Schwartz, Roy and Smith, Noah A. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194. Association for Computational Linguistics.

William B Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Chengyu Dong, Liyuan Liu, and Jingbo Shang. 2022. SoTeacher: Toward Student-oriented Teacher Network Training for Knowledge Distillation.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Mark Everingham, Luc Van Gool, CKI Williams, John Winn, and Andrew Zisserman. 2012. The PASCAL Visual Object Classes Challenge 2012 (VOC2012).

Alireza Furutanpey, Johanna Barzen, Marvin Bechtold, Schahram Dustdar, Frank Leymann, Philipp Raith, and Felix Truger. 2023a. Architectural Vision for Quantum Computing in the Edge-Cloud Continuum. *arXiv preprint arXiv:2305.05238*.

Alireza Furutanpey, Philipp Raith, and Schahram Dustdar. 2023b. FrankenSplit: Saliency Guided Neural Feature Compression with Shallow Variational Bottleneck Injection. *arXiv preprint arXiv:2302.10681*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew E Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The Third PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, and Sourab Mangrulkar. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Odd Erik Gundersen. 2019. Standing on the Feet of Giants - Reproducibility in AI. *AI Magazine*, 40(4):9–23.

Odd Erik Gundersen, Yolanda Gil, and David W. Aha. 2018. On Reproducible AI: Towards Reproducible Research, Open Science, and Digital Scholarship in AI Publications. *AI Magazine*, 39(3):56–68.

Cong Guo, Chen Zhang, Jingwen Leng, Zihan Liu, Fan Yang, Yunxin Liu, Minyi Guo, and Yuhao Zhu. 2022. ANT: Exploiting Adaptive Numerical Data Type for Low-bit Deep Neural Network Quantization. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1414–1433. IEEE.

Shivanshu Gupta, Yoshitomo Matsubara, Ankit Chadha, and Alessandro Moschitti. 2023. Cross-lingual knowledge distillation for answer sentence selection in low-resource languages. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7259–7272.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pages 785–794.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Tong He, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan. 2019. Knowledge Adaptation for Efficient Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 578–587.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. 2019. Knowledge Transfer via Distillation of Activation Boundaries Formed by Hidden Neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3779–3787.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the Knowledge in a Neural Network. In *Deep Learning and Representation Learning Workshop: NIPS 2014*.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Chris Kamphuis, Arjen P de Vries, Leonid Boytsov, and Jimmy Lin. 2020. Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 28–34. Springer.

Jangho Kim, SeongUk Park, and Nojun Kwak. 2018. Paraphrasing Complex Network: Network Compression via Factor Transfer. In *Advances in Neural Information Processing Systems*, pages 2760–2769.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Third International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Heejun Lee, Minki Kang, Youngwan Lee, and Sung Ju Hwang. 2022. Sparse Token Transformer with Attention Back Tracking. In *The Eleventh International Conference on Learning Representations*.

Seungjae Ryan Lee and Seungmin Lee. 2023. [Re] Pure Noise to the Rescue of Insufficient Data. In *ML Reproducibility Challenge 2022*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 552–561.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A Community Library for Natural Language Processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.

Wei Li, Shitong Shao, Weiyan Liu, Ziming Qiu, Zhihao Zhu, and Wei Huan. 2022a. What Role Does Data Augmentation Play in Knowledge Distillation? In *Proceedings of the Asian Conference on Computer Vision*, pages 2204–2220.

Yulong Li, Martin Franz, Md Arafat Sultan, Bhavani Iyer, Young-Suk Lee, and Avirup Sil. 2022b. Learning Cross-Lingual IR from an English Retriever. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4428–4436.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362.

Sihao Lin, Hongwei Xie, Bing Wang, Kaicheng Yu, Xiaojun Chang, Xiaodan Liang, and Gang Wang. 2022. Knowledge Distillation via the Target-aware Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10915–10924.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Li Liu, Qingle Huang, Sihao Lin, Hongwei Xie, Bing Wang, Xiaojun Chang, and Xiaodan Liang. 2021. Exploring Inter-Channel Correlation for Diversity-preserved Knowledge Distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8271–8280.

Daniel Lopresti and George Nagy. 2021. Reproducibility: Evaluating the Evaluations. In *International Workshop on Reproducible Research in Pattern Recognition*, pages 12–23. Springer.

Michael S Matena and Colin A Raffel. 2022. Merging Models with Fisher-Weighted Averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

Yoshitomo Matsubara. 2021. torchdistill: A Modular, Configuration-Driven Framework for Knowledge Distillation. In *International Workshop on Reproducible Research in Pattern Recognition*, pages 24–44. Springer.

Yoshitomo Matsubara, Davide Callegaro, Sameer Singh, Marco Levorato, and Francesco Restuccia. 2022a. BottleFit: Learning Compressed Representations in Deep Neural Networks for Effective and Efficient Split Computing. *arXiv preprint arXiv:2201.02693*.

Yoshitomo Matsubara, Luca Soldaini, Eric Lind, and Alessandro Moschitti. 2022b. Ensemble Transformer for Efficient and Accurate Ranking Tasks: an Application to Question Answering Systems. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7259–7272.

Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, and Stephan Mandt. 2022c. Supervised Compression for Resource-Constrained Edge Computing Systems. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2685–2695.

Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, and Stephan Mandt. 2023. SC2 Benchmark: Supervised Compression for Split Computing. *Transactions on Machine Learning Research.*

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. *Advances in Neural Information Processing Systems*, 26.

Roy Miles and Krystian Mikolajczyk. 2023. A closer look at the training dynamics of knowledge distillation. *arXiv preprint arXiv:2303.11098.*

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved Knowledge Distillation via Teacher Assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5191–5198.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational Knowledge Distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3967–3976.

Nikolaos Passalis and Anastasios Tefas. 2018. Learning Deep Representations with Probabilistic Knowledge Transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–284.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. 2021. Improving Reproducibility in Machine Learning Research(A Report from the NeurIPS 2019 Reproducibility Program). *The Journal of Machine Learning Research*, 22(1):7459–7478.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Nils Reimers and Iryna Gurevych. 2020. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in neural information processing systems*, pages 91–99.

Anna Rogers, Timothy Baldwin, and Kobi Leins. 2021. 'Just What do You Think You're Doing, Dave?'A Checklist for Responsible Data Use in NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4821–4833.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, 27.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. Contrastive Representation Distillation. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in neural information processing systems*, 30.

Leandro Von Werra, Lewis Tunstall, Abhishek Thakur, Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, and Helen Ngo. 2022. Evaluate & Evaluation on the Hub: Better Best Practices for Data and Model Measurements. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 128–136.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Amanpreet Singh, and Samuel Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. 2020. Knowledge Distillation Meets Self-supervision. In *European Conference on Computer Vision*, pages 588–604. Springer.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–20.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically Examining the "Neural Hype": Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1129–1132.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press.

Youcai Zhang, Zhonghao Lan, Yuchen Dai, Fangao Zeng, Yan Bai, Jie Chang, and Yichen Wei. 2020. Prime-Aware Adaptive Distillation. In *The European Conference on Computer Vision (ECCV)*.