

# Best practice for text classification with distillation-

## Part 1/3

In recent years, increasingly large Transformer-based models such as BERT have demonstrated remarkable state-of-the-art (SoTA) performance in many Natural Language Processing (NLP) tasks and have become the de-facto standard.

However, there is no free lunch (actually, wait till you finish this blog series). These models are extremely inefficient and require massive computational resources and large amounts of data for training and deploying. This severely hinders the scalability and deployment of NLP-based systems across the industry.

I have always been fascinated by robustness and efficiency in NLP deployed in production. So I decided to write a blog series that will provide some practical tips and code samples for deploying and adapting large SoTA transformer models.

In the first three blogs I will focus on model distillation for text classification. Model distillation is a very powerful pruning technique and in many use-cases it yields significant speed-up and memory size reduction. It's generally considered more suitable for advanced users because it is relatively hard to implement, its performance is unpredictable, and the inherent mechanism is obscure. I will try to show just the opposite by providing a few real use-case examples with simple code snippets and intuitive explanations for the effectiveness of model distillation.

Let's begin.

Suppose you are a data scientist in a top enterprise company and your task is to classify social media tweets and deliver SoTA models into production. You were also warned that your model must be very efficient and you will pay a high cost for any extra million parameters. You will probably start by collecting sufficient data (several hundreds or thousands of labeled samples) and then you'll compare several ML/DL models and Transformer models for maximum accuracy at minimum cost (i.e., minimal model size).

To demonstrate the accuracy achieved by different model types, I chose an emotion classification dataset called [Emotion](#) that consists of Twitter posts labeled with any of six basic emotion categories: sadness, disgust, anger, joy, surprise, and fear. The data consists of 16K training samples and 2K test samples and is available on the HuggingFace Datasets Hub. A code example for the following steps is available [here](#).

### **1st step: Set a baseline using a logistic regression model (Tf-Idf based)**

After performing the first step, we get:

Accuracy = 86.1%

That is our baseline result.

## 2nd step: Set a deep learning baseline

Next, we try a simple MLP model. The model architecture is very basic and includes an input dim size of 5000 (max word vocabulary size), output dim size of 16, average pooling layer, and softmax (in total ~80K parameters).

We get:





Accuracy = 86%

The accuracy is similar to the accuracy of the logistic regression but the model is more efficient and more dense.

## 3rd step: Transformer models

As NLP veterans and practiced users of HuggingFace and their amazing “transformers” library, we try a few popular SoTA transformer models.

We get:



Model	Accuracy	#Parameters	Implementation Source
BERT-base	92.4	110M	 
RoBERTa	92.7	125M	 
T5	93.5	11B	<a href="#">t5-base-finetuned-emotion</a>

Awesome! Accuracy shot sky high. But 110M parameters is way above our computational budget, and IT will hit the ceiling when they hear about the 11B model (:

## 4th step: DistilBERT

So let's try instead one of the more popular models like DistilBERT or DistilRoBERTa, released by HuggingFace, which is half the size yet double the speed compared to the BERT-base model.

We get:

Model	Accuracy	#Parameters	Implementation Source
DistilBERT	91.5	66M	 
DistilRoBERTa	92.3	82M	<a href="#">Elvis Saravia</a>

Nice. DistilBERT's accuracy dropped by less than 1 percent compared with the BERT-base model, and our model is much smaller.

Is that it? Are we done? Should we go to production with this model and pay the computing cost for the 67M parameters?

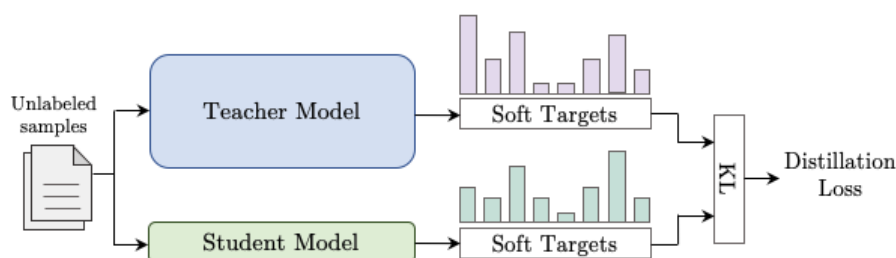
Or can we do better? Can we use less than 1M or 100K parameters while maintaining minimal accuracy loss (<1% loss)?

The answer is a resounding “Yes!” (to a degree that varies with your dataset quality and task at hand). We will harness to our purpose knowledge distillation and additional data either by utilization of data augmentation techniques or sampled from your in-domain unlabeled dataset.

As you know, BERT is a pre-trained language model trained for the Mask Language Model task. In our case we are interested only in emotion classification, and previous research shows that fine-tuned BERT parameters are over-parameterized for domain specific tasks ([Kovaleva et al., 2019](#)).

This is where Knowledge Distillation comes in.

Knowledge distillation (KD) to much simpler architectures ([Tang et al., 2019](#); [Wasserblat et al., 2020](#)) showed promising results for reducing model size and computational load while preserving much of the original model's performance. A typical model distillation setup includes two stages. In the first stage, a large, cumbersome and accurate teacher neural network is trained for a specific downstream task. In the second stage, shown in the following figure, a smaller and simpler student model that is more practical for deployment in environments with limited resources, is trained to mimic the behavior of the teacher model.



Code disclaimer: To make system super easy, we only use a single distillation loss which is generated for each training batch by calculating the Kullback–Leibler (KL) distance between the target predictions that are produced by the student and teacher models. We didn't notice any performance loss when deploying MSE between soft targets (logits), nor when we employed temperature as in the original distillation paper ([Hinton et al., 2015](#)). Distillation setups are usually cumbersome due to the use of different loss types for labeled and unlabeled data, whereas, our implementation is simpler and friendlier since it only uses a single loss (KL loss) for both types of data.

### 5th step: Distill RoBERTa to a simpler student

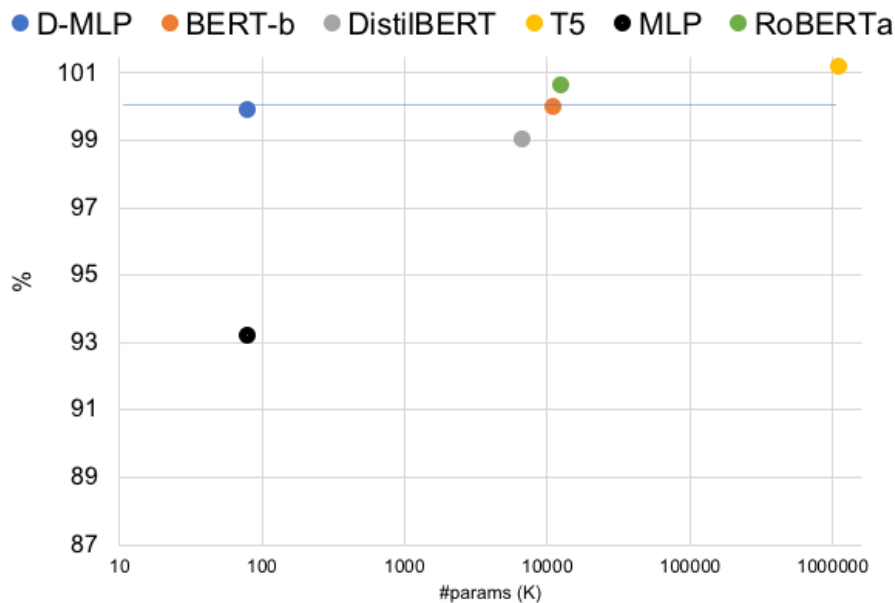
In our case we distill RoBERTa's knowledge into our simple MLP model.

Here are the results:

Model	Accuracy	#Parameters	Implementation Source
MLP	86.4	80K	Keras
Distilled MLP	92.3	80K	Keras

Wow!!! Surprisingly not bad at all, with accuracy even higher than DistilBERT and on-par with BERT!

The following figure summarizes the results that we achieved so far for **Emotion** in terms of (model acc./BERT-base acc.)% vs. model size.



Could this really be true? We distilled RoBERTa's knowledge into our tiny model with almost no loss. We benefitted from high transformer model performance with very little cost to pay (and made IT very happy).

By now you probably have many questions, such as:

- Does this “trick” hold for **any** text-classification sub-task?
- If not, when does it work?
- How would I choose the best student for my dataset?
- What is the intuition behind this behavior?

I'll try to answer these questions and provide a bit of intuition in the following posts.