Supplementary Material for SqueezeBERT

A Overview of GLUE tasks

We evaluated the SqueezeBERT model on the GLUE benchmark. This benchmark includes 8 text-classification tasks, which include sentiment analysis, entailment, and grammatical correctness: MNLI (Williams et al., 2018), QQP (Quora, 2017; Chen et al., 2018), QNLI, SST-2 (Socher et al., 2013), CoLA (Warstadt et al., 2019), MRPC (Dolan and Brockett, 2005), RTE (Bentivogli et al., 2009), and WNLI (Levesque et al., 2012). The GLUE benchmark also includes one text-regression task, STS-B (Cer et al., 2017), which involves scoring the level of similarity between two sentences.

B Training Hardware

We do all pretraining and finetuning on an 8-GPU server without multi-server distributed training. Our server has 8 NVIDIA Titan RTX GPUs, an Intel Xeon Gold 6130 64-core CPU, and 256GB of RAM. On this 8-GPU server, the SqueezeBERT model can be reproduced from scratch in approximately 5 days: 4 days for pretraining, and then under one day for finetuning on all GLUE tasks with the optimal hyperparameters discovered by our hyperparameter search (see Section D).

C Training Software

Our PyTorch-based training and inference code draws heavily on the HuggingFace Transformers (Wolf et al., 2019) and NVIDIA Deep Learning Examples (NVIDIA, 2020b) repositories. We perform pretraining using 8 GPUs with 16-bit floating-point math, and we use the O2 optimization level in the NVIDIA Apex mixed-precision training primitives (NVIDIA, 2020a). We perform finetuning on a single GPU with 32-bit floating-point math, and we concurrently run multiple finetuning tasks across the 8-GPU machine.

D Details of hyperparameter search during finetuning on GLUE tasks

We now present more details on the hyperparameter search approach that we used for training SqueezeBERT with bells and whistles. In Table 4, we present the space of possible hyperparameters over which we performed a grid-search. Note that the time to finetune the model using one set of hyperparameters varies significantly depending on the GLUE task, from 15 minutes for small datasets like RTE, to 14 hours for MNLI. For smaller datasets that require less training (e.g. RTE and MRPC), we use a broader search space and more epochs. And, for larger datasets (e.g. MNLI and QQP), we use a more narrow search space with fewer epochs.

Now, in Table 5, we present the best hyperparameters found in our search. We observe two interesting phenomena on this table. The first is regarding the use of distillation. Recall from Section 4.2.2 that α is the hyperparameter that sets the weighting between the teacher logits and the ground-truth for distillation. When $\alpha = 1.0$, the teacher logits are ignored, and thus distillation is disabled. So, it is interesting to note that on three of the eight GLUE tasks in Table 5 (MNLI, QNLI, and CoLA), distillation did not produce superior results over non-distillation finetuning. The second interesting phenomenon in this table is that maximum accuracy was not necessarily achieved on final epoch. For example, on QNLI, SqueezeBERT converged to its maximum development-set accuracy after just two epochs.

Table 4: Our hyperparameter search space.

Hyperparameter	MNLI, QQP, QNLI, STS-2	STS-B, MRPC, RTE	CoLA
α	[0.8, 0.9, 1.0]	[0.8, 0.9, 1.0]	[0.8, 0.9, 1.0]
Learning Rate	[1e-05, 2e-05, 3e-05, 4e-05]	[1e-05, 2e-05, 3e-05, 4e-05, 5e-05]	[1e-05, 2e-05, 3e-05, 4e-05, 5e-05]
Encoder Dropout	[0.0, 0.1]	[0.0, 0.1]	[0.0, 0.1]
Final Dropout	[0.1, 0.2]	[0.1, 0.2]	[0.0, 0.1]
Epochs	5	10	20
Batch Size	16	16	[16, 32, 48]

Table 5: Selected hyperparameters used for training SqueezeBERT with bells-and-whistles.

Hyperparameter	MNLI-m	MNLI-mm	QQP	QNLI	STS-2	CoLA	STS-B	MRPC	RTE
α	1.0	1.0	0.8	1.0	0.8	1.0	0.8	0.9	0.8
Learning Rate	3e-05	3e-05	4e-05	3e-05	3e-05	1e-05	4e-05	3e-05	3e-05
Encoder Dropout	0.0	0.0	0.1	0.0	0.1	0.0	0.1	0.1	0.1
Final Dropout	0.1	0.1	0.1	0.1	0.1	0.0	0.1	0.1	0.1
Epochs	4	4	5	2	5	6	10	9	3
Batch Size	16	16	16	16	16	32	16	16	16