Maximal Multiverse Learning for Promoting Cross-Task Generalization of Fine-Tuned Language Models

Itzik Malkiel
Tel Aviv University
itzik.malkiel@gmail.com

Lior Wolf
Tel Aviv University
wolf@cs.tau.ac.il

Abstract

Language modeling with BERT consists of two phases of (i) unsupervised pre-training on unlabeled text, and (ii) fine-tuning for a specific supervised task. We present a method that leverages the second phase to its fullest, by applying an extensive number of parallel classifier heads, which are enforced to be orthogonal, while adaptively eliminating the weaker heads during training. We conduct an extensive inter- and intra-dataset evaluation, showing that our method improves the generalization ability of BERT, sometimes leading to a +9% gain in accuracy. These results highlight the importance of a proper fine-tuning procedure, especially for relatively smaller-sized datasets. Our code is attached as supplementary.

1 Introduction

Recently, there has been an increasing number of studies suggesting the use of general language models, for improving natural language processing tasks (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Howard and Ruder, 2018). Among the most promising techniques, the unsupervised pretraining approach (Dai and Le, 2015; Radford et al., 2018) has emerged as a very successful method, that achieves state-of-the-art results on many language tasks, including sentiment analysis (Socher et al., 2013), natural language inference (Williams et al., 2017) and similarity and paraphrasing tasks (Dolan and Brockett, 2005; Cer et al., 2017). This approach incorporates a two-phase training procedure. The first phase utilizes an unsupervised training of a general language model on a large corpus. The second phase applies supervision to fine-tune the model for a given task.

More recently, unsupervised pretraining models such as BERT (Devlin et al., 2018), XLNET (Yang et al., 2019) and RoBERTa (Liu et al., 2019), have achieved unprecedented performance. For example, in the GLUE benchmark (Wang et al., 2018), BERT (Devlin et al., 2018) was reported to achieve performance that exceeds human level on a few different datasets, such as QNLI (Rajpurkar et al., 2016), QQP (Chen et al., 2018) and MRPC (Dolan and Brockett, 2005). However, despite the great progress achieved by these task-specific and dataset-specific models, it is not yet clear how well they can generalize to different tasks, how well they generalize when evaluating the same task on different datasets, and how to improve this generalization ability.

In our work, we extend the multiverse method of (Littwin and Wolf, 2016), which was shown to improve transfer learning in the computer vision task of face recognition and on the CIFAR-100 small image recognition dataset. The multiverse loss generalizes the cross entropy loss, by simultaneously training multiple linear classification heads to perform the same task. In order to prevent multiple copies of the same classifier, in the multiverse scheme, each classifier is mutually orthogonal to the rest of classifiers. The number of multiverse heads used was limited, never more than seven and typically set to five.

We propose a novel fine-tuning procedure for enhancing the generalization ability of the recent unsupervised pretrained language models, by employing a large number of multiverse heads. The essence of our technique is as follows: given a pretrained language model and a downstream task with labeled data, we fine-tune the model using a maximal number of multiverse classifiers. The fine-tuning goal is to both minimize the task loss and an orthogonality loss applied to the classification heads. When enforcing orthogonality hinders the classifiers' performance, we detect and eliminate the less effective classification heads.

The technique, therefore, preserves a maximal set of classifiers, which is comprised of the best performing ones. By maintaining this maximal subset during training, our method leverages multiverse loss to its fullest. Hence, we name our method Maximal Multiverse Learning (MML).

Our contributions are as follows: (1) we present MML, a general training procedure to improve the transferability of neural models. (2) we apply MML on BERT and report its performance on various datasets. (3) we propose a set of cross dataset evaluations using common NLP benchmarks, demonstrating the effectiveness of MML, in comparison to regular BERT fine-tuning and to alternative regularization techniques.

2 Related work

Many recent breakthroughs in NLP employ unsupervised pretraining of language models. The different variants can be categorized into two main approaches: (1) feature-based models, such as (Peters et al., 2018) and (2) fine-tuning models, such as (Devlin et al., 2018; Liu et al., 2019; Yang et al., 2019). The former technique utilizes a language neural-based model as a feature extractor. The extracted features may be used for the training of another model, receiving the extracted features as input. The second approach utilizes a similar pre-trained model, but fine-tunes it in an end-to-end manner to specialize on a given task. During the fine-tuning phase, all of the parameters of the model are updated and a relatively small number of parameters are trained from scratch.

The usage of multiple classifiers can be found in few places in the literature. In GoogLeNet (Szegedy et al., 2015), the authors use multiple classifier heads in different places in the model architecture. The additional classifiers led to better propagation of the gradients during training. However, with the advent of better conditioning and normalization methods, as well as with the modern introduction of skip connections in architectures such as the ResNet (He et al., 2016), the practice of adding intermediate branches, for the sake of introducing an auxiliary loss at lower layers, was mostly abandoned.

The multiverse loss was shown to promote transfer learning and to lead to a low-dimensional representation in the penultimate layer (Littwin and Wolf, 2016). However, the current literature does not present any methodological way to select the number of multiverse heads and the idea was only applied for a handful of parallel classifiers. In MML, hundreds of multiverse heads are used, leading to a tradeoff between the classifier accuracy and the orthogonality constraint. MML balances the two terms by pruning, during training, the under-performing heads.

3 Method

Let $\mathscr{W} = \{w_i\}_{i=1}^{w}$ be the vocabulary of tokens in a given language. Let Y be the set of all possible sentences generated by $\mathscr{W}$, including the empty sentence. A language model $M : Y \times Y \to \mathbb{R}^d$ receives a pair of elements from Y and returns a vectors of d dimensions. Given a dataset with n training samples, $s_1 \dots s_n \in Y \times Y$, each associated with a label $y_i \in [1 \dots c]$, we denote the coding vector of each sample by $d_i := M(s_i)$. As a concrete example, for the BERT model, d_i is the latent embedding of the CLS token.

Common language models employ a classifier $C : \mathbb{R}^d \to \mathbb{R}^c$ that projects the coding vectors $d_i \in \mathbb{R}^d$ by a $d \times c$ matrix, $F_{d\times c} = [f_1, \dots, f_c]$, $(f_i \in \mathbb{R}^d)$, and then adds a bias term $b \in \mathbb{R}^c$:

$$C(d_i) = d_i^T F_{d\times c} + b \tag{1}$$

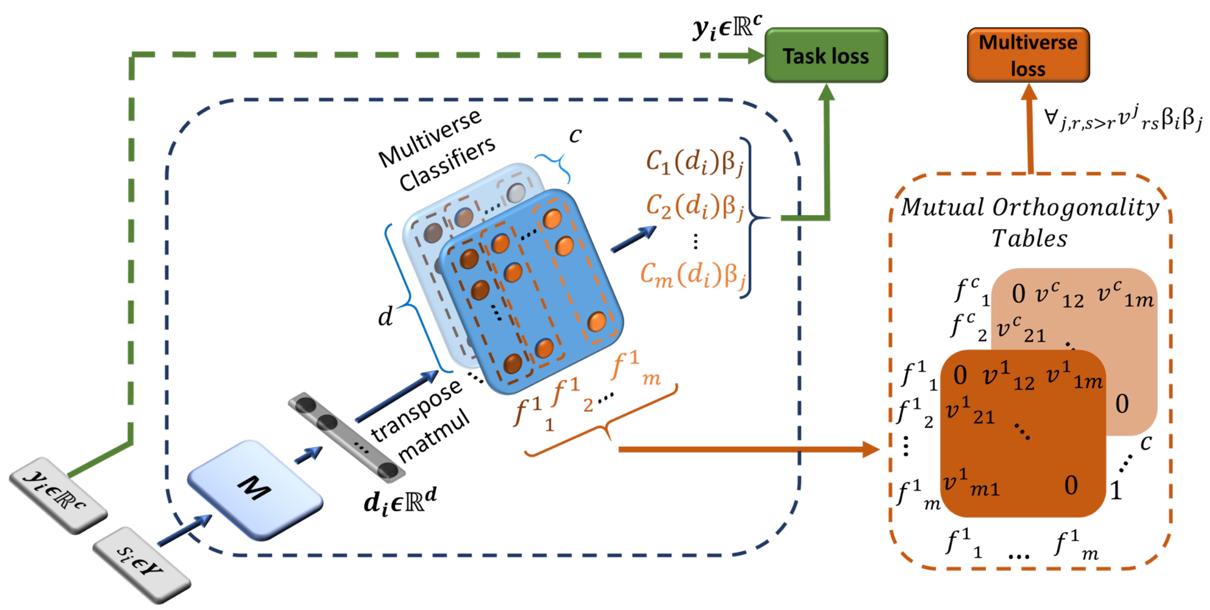
The output of C is a logit vector, and pseudo-probabilities are obtained by applying softmax $p_i(y_i) = e^{C(d_i)_{y_i}} / \sum_{j=1}^{c} e^{C(d_i)_j}$.

Different from the single-classifier models, our model utilizes a multiverse classifier $\mathscr{C} : \mathbb{R}^d \to \mathbb{R}^{c\times m}$ defined as $\mathscr{C}(d_i) = (C_1(d_i), \dots, C_m(d_i))$, where m is a multiplicity parameter, $\{C_j : \mathbb{R}^d \to \mathbb{R}^c\}_{j=1}^{m}$ are parallel classifiers, each with different weights, applying the same function as Eq. 1. :

$$C_j(d_i) = d_i^T F_{d\times c}^j + b_j \tag{2}$$

Additionally, we will define $B = \{\beta_j \in \{0,1\}\}_{j=1}^{m}$ as a set of binary scalars. Each classifier head C_j will be associated with a different binary scalar β_j, which is set during training. In our experiments, we set m to be equal to the coding vector size d, which entails a full rank of active multiverse classifiers at the beginning of the training.

The loss function is composed of two components, the task loss and the multiverse loss. Defining active multiverse classifiers as those that are associated with a value $\beta_j = 1$, the task loss optimizes the performance of all active multiverse



$y_i \epsilon \mathbb{R}^c$
Task loss
Multiverse loss
$\forall_{j,r,s>r} v^j{}_{rs} \beta_i \beta_j$
Multiverse Classifiers
c
d
$C_1(d_i)\beta_j$
$C_2(d_i)\beta_j$
$C_m(d_i)\beta_j$
Mutual Orthogonality Tables
transpose matmul
M
$d_i \epsilon \mathbb{R}^d$
$y_i \epsilon \mathbb{R}^c$
$s_i \epsilon \mathcal{Y}$
$f^1{}_1$ $f^1{}_2$ … $f^1{}_m$

Figure 1: A schematic illustration of the MML model. The task loss comprises a loss-term for each multiverse classifier, using the given labels of the task at hand. The mutual orthogonality tables hold the absolute value of the dot product calculated between the weights of all classifiers, across the different classes. Since orthogonality of a classifier with itself is ignored, we set the diagonal to 0. Following multiverse loss definition and since orthogonality is symmetric, only half of each table value is passed to the multiverse loss.

Algorithm 1 MML training. The MeanShift function returns the center of the clusters. Params: $K = 1000$, $\gamma = 0.99$, $Threshold = 5$, $\alpha = 2 \cdot e^{-5}$, $\lambda = 0.005$.

```
$\forall 1 \le j \le m : \beta_j \leftarrow 1, a_j \leftarrow 0$
for $step = 1, 2, \ldots$ do
    Sample a minibatch $\{(s_i, y_i)\}_{i=1}^{t}$
    $MC_\theta \leftarrow \nabla_\theta \left[ \lambda \cdot \mathcal{L}_{mv}(\mathscr{C}, B) + \frac{1}{t}\sum_{i=1}^{t} \mathcal{L}_{task}\left(M, \mathscr{C}, B, \{(s_i, y_i)\}_{i=1}^{t}\right) \right]$
    for $1 \le j \le m$ do
        $a_j \leftarrow (1 - \gamma) \cdot a_j + \left(\gamma \cdot \frac{1}{t}\sum_{i=1}^{t} \mathcal{L}_{task}\left(M, \mathscr{C}, B, \{(s_i, y_i)\}_{i=1}^{t}\right)\right)$
    $\theta \leftarrow \theta + \text{Adam}(\theta, MC_\theta, \alpha)$
    if $step \% K = 0$ and $\sum_j \beta_j \ge Threshold$ then
        clusters $\leftarrow$ MeanShift$\left(\{a_j | \beta_j = 1\}_{j=1}^{m}\right)$
        if $|clusters| \ge 2$ then
            for $1 \le j \le m$ do
                $\beta_j \leftarrow 0$, if $a_j \notin min(clusters)$
```

classifiers, each independently, using the supervision obtained by the given labels. The multiverse loss soft-enforces orthogonality among the active classifiers. Its purpose is to regularize the model by encouraging M to produce coding vectors that are robust enough to be effective for a large number of orthogonal classifiers.

As mentioned earlier, each classifier C_j is associated with a binary value β_j, which controls the applicability of the classifier and is configured during training. Under the context of the loss function, setting β_j to 0 would eliminate the impact of the j^{th} classifier head C_j for both the task loss and multiverse loss.

For a multi-class classification task, we apply the following task loss:

$$\mathcal{L}_{task} = -\Sigma_{i=1}^{n} \Sigma_{j=1}^{m} \mathcal{L}_{cce}^{j,i} \beta_j \tag{3}$$

where n is the number of training samples, and $\mathcal{L}_{cce}^{j,i} = y_i log(C_j(M(s_i))_{y_i})$ is the cross entropy loss. For a binary classification task we set $\mathscr{C} : \mathbb{R}^d \to \mathbb{R}^{2\times m}$ and use the same loss from Eq. 3. For a regression task, we replace $\mathcal{L}_{cce}^{j}$ with $\mathcal{L}_{L2}^{j,i} = \|y_i - C_j(M(s_i))\|_2^2$.

The second loss term enforces orthogonality between the set of classifiers, for each class separately, using the multiverse loss:

$$\mathcal{L}_{mv} = \Sigma_{j,r,s>r} \left| f_j^{r\top} f_j^{s} \beta_r \beta_s \right| \tag{4}$$

where f_j^r is the jth column of the weight matrix that corresponds to classifier C_r.

The total loss $\mathcal{L}_{total}$ is defined as: $\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda \mathcal{L}_{mv}$. Similar to (Littwin and Wolf, 2016), we set $\lambda = 0.005$, throughout all of our experiments. The MML model is illustrated in Fig. 1.

The training algorithm begins with an initialization of the aggregated model $(M, \mathscr{C}, B)$. M may be initialized by any pre-trained general language model. The multiverse classifiers are randomly initialized from scratch, and all classifiers are initially activated, by setting $\beta_j = 1$ for $\forall \beta_j \in B$.

During training, we track the performance of each multiverse classifier separately. Every K steps, we search for a subset of the top-performing classifiers. When we find such a subset, we eliminate the less performing classifiers by setting their corresponding βs to 0.

In order to detect the top-performing classifiers, we calculate a moving average variable a_j for each multiverse classifier. Specifically, a_j holds the moving average of the task loss value $\mathcal{L}^j_{task}$ associated with classification head C_j. a_j is being updated for every training step, using the moving average momentum constant of 0.99.

Every K steps, we run the MeanShift algorithm (Comaniciu and Meer, 2002) on the set $\{a_j | \beta_j = 1\}$ to obtains the modes of the underlying distribution. MeanShift is a clustering algorithm that reveals the number of clusters in a given data, and retrieves the corresponding centroid for each detected cluster. In our case, MeanShift is applied on 1D data and by utilizing it, we identify the subset of top-performing multiverse heads as the cluster associated with the minimal centroid value. Next, we eliminate the rest of the multiverse heads, by setting their corresponding β to 0. This adaptive elimination is stopped, when we reach a minimal number of active heads, see Alg. 1.

At inference, we use the active multiverse heads to retrieve predictions. Specifically, given a sample s_i, we calculate the logits $\hat{y}$ as:

$$\hat{y} := \frac{\sum_{j=1}^{m} C_j(M(s_i)) \cdot \beta_j}{\sum_{j=1}^{m} \beta_j} \qquad (5)$$

for classification tasks, we apply the softmax function on $\hat{y}$, and return its output. For regression tasks, we simply return $\hat{y}$.

4 Results

In this study, we evaluate MML, applied on a pre-trained BERT (Devlin et al., 2018) model, using nine NLP datasets, while employing two different settings: (1) fine-tuning on different downstream tasks from the GLUE benchmark (Wang et al., 2018), and (2) cross dataset evaluations for different datasets of the same or similar tasks. For the first, we fine-tune MML on each dataset separately, and evaluate its performance on the development set and the test set of the same dataset. For the second, we evaluate our fine-tuned MML models on the train and development set of other datasets within the same task category. This allows us to study the generalization level of all models, across different datasets. Additionally, we perform an ablation study and report empirical results that showcase the efficacy of MML, compared to other multiverse schemes and to a BERT model with a higher dropout rate.

We adopt eight datasets from the GLUE benchmark (Wang et al., 2018) and one extra dataset supporting the task of Natural Language Inference (NLI). The datasets can be arranged to form the following three categories: (1) Inference tasks: this category incorporates four datasets of natural language inference: RTE, MNLI, SNLI and QNLI. RTE and QNLI are binary classification tasks, whereas MNLI and SNLI are multilabel classification tasks (which possess an additional "neutral" label). (2) Similarity and paraphrasing tasks: this category includes three datasets: MRPC, QQP, STS-B. MRPC and QQP are binary classifications tasks, while STS-B is a regression task with labels annotating the level of similarity between each sentence pair. (3) Misc. datasets: this category composed of two datasets that cannot be used for the cross dataset evaluation, due to the lack of commonality between their tasks: CoLA and SST-2. We refer the reader to the appendix for more details on the nine datasets.

4.1 Evaluation on GLUE datasets

We evaluated MML on the eight different datasets from the GLUE benchmark, and compared to BERT (Devlin et al., 2018). Each model was trained and evaluated on a single dataset. Development and test set performance are being reported for each model. In addition, we conduct an ablation analysis for our method, presenting the importance of our Maximal Multiverse Learning, which allows the training to adapt the number of multiverse classifiers to each dataset. The first ablation experiment disables the classifier elimination step during training and utilizes the same MML architecture with a fixed number of heads. The second ablation also disables the multiverse loss, leaving the training to optimize an ensemble of classifiers, without enforcing the or-

	Model	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
		392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
development set	BERT	86.6/-	91.3	92.3	93.2	60.6	90.0	88.0	70.4	84.0
	MV-5	87.0/-	91.4	92.2	**94.0**	64.3	**91.1**	88.0	75.4	85.4
	MV-1024	86.2/-	91.4	92.2	93.2	57.9	90.6	**89.7**	**80.1**	85.2
	BERT-Drop	87.0/-	91.6	**93.0**	92.9	62.3	90.7	87.5	76.5	85.2
	BERT-Ensemble	86.6/-	91.4	92.3	93.2	60.9	90.2	88.0	70.4	84.1
	MML	**87.2**/-	**91.7**	**93.0**	**94.0**	**64.5**	**91.1**	89.0	**80.1**	**86.3**
test set	BERT	86.7/85.9	89.3	92.7	**94.9**	**60.5**	86.5	85.4	70.1	83.2
	MV-5	86.4 /85.9	88.9	92.2	94.1	56.9	87.4	86.3	70.6	82.8
	MV-1024	87.0/85.9	89.1	92.2	93.8	54.8	86.8	86.1	**74.2**	82.9
	BERT-Drop	86.6/86.0	89.3	**92.8**	94.1	56.1	87.9	85.3	74.1	83.2
	BERT-Ensemble	86.6/86.0	89.2	92.3	94.1	60.1	86.4	86.1	70.4	83.1
	MML	**87.0/86.0**	**89.4**	92.6	94.6	58.6	**88.1**	**86.7**	**74.2**	**83.8**
	MML #heads	19	14	23	979	31	45	913	1024	-

Table 1: Results on GLUE benchmarks (Wang et al., 2018). BERT results taken from (Devlin et al., 2018). Accuracy scores are reported for all datasets, except STS-B, for which Spearman Correlation is reported. For MNLI, accuracy scores are reported for both matched and mismatched test sets. The last row exhibits the number of active multiverse heads of the converged MML model. For example, for MRPC, our MML model used 913 active multiverse heads, while for MNLI, it maintained 19.

thogonality constraint.

The BERT model used is the BERT-Large model (Devlin et al., 2018). It contains 24 attention layers, each with 16 attention heads with a hidden layer of size $d = 1024$ dimensions. The model was pre-trained using sentence pairs, to both reconstruct masked words and to predict whether the pairs are consecutive. BERT's fine-tuning for downstream tasks employs supervision obtained by the given labels of each dataset.

MML utilizes the same pre-trained BERT-Large model and is initialized with $m = d = 1024$ active multiverse classifiers. During training, the model converges to a smaller number of multiverse classifiers. The number of active multiverse classifiers of each model is presented in last row of Tab. 1.

Tab. 1 presents the results for six models: (1) BERT (as a baseline), (2) MML, (3-4) two multiverse models utilizing a fixed number of multiverse classifiers, with 5 and 1024 classifiers, respectively, (5) BERT-Drop, a BERT model that was fine-tuned with a 30% dropout rate on the CLS embedding vector and which provides a baseline with additional regularization, (6) BERT-Ensemble, a BERT model with 1024 heads (similar to MV-1024 without the multiverse loss).

As can be seen in the table, compared to BERT, MML yields better results by a sizeable margin on the test set of five out of eight datasets. The largest gains were reported in the relatively smaller-sized dataests, such as RTE, MRPC and STS-B, for whom MML yields an absolute improvement of 4.1%, 1.3%, 1.6%, respectively. This can be attributed to the ability of MML to encourage a more robust learning. On the rest of the datasets, MML yields similar performance on the test. On the development set, MML outperforms BERT on all datasets, sometimes by a large margin. Specifically, for RTE, MML yields an improvement of almost 10%.

The ablation models MV-5 and MV-1024, utilize a fixed number of multiverse heads during the entire training. We have found that this hyper parameter can be crucial for the model's convergence, and when not initialized properly, may significantly reduce performance for the given task in hand. Specifically, for the CoLA dataset, MV-1024 and MV-5 yield a relative performance gap of more than 11%, in favor of MV-5, while in RTE, there is a gap of 6.2% in favor of MV-1024. When comparing both MV-5 and MV-1024 to MML, MML produces better or similar performance on the development set of all datasets. More specifically, on RTE and MRPC, MML yields similar performance as in MV-1024, and outperforms it on all the other six datasets. Compared to MV-5, MML yields better performance, by a sizeable margin, on four datasets out of eight, and produces similar performance on the rest.

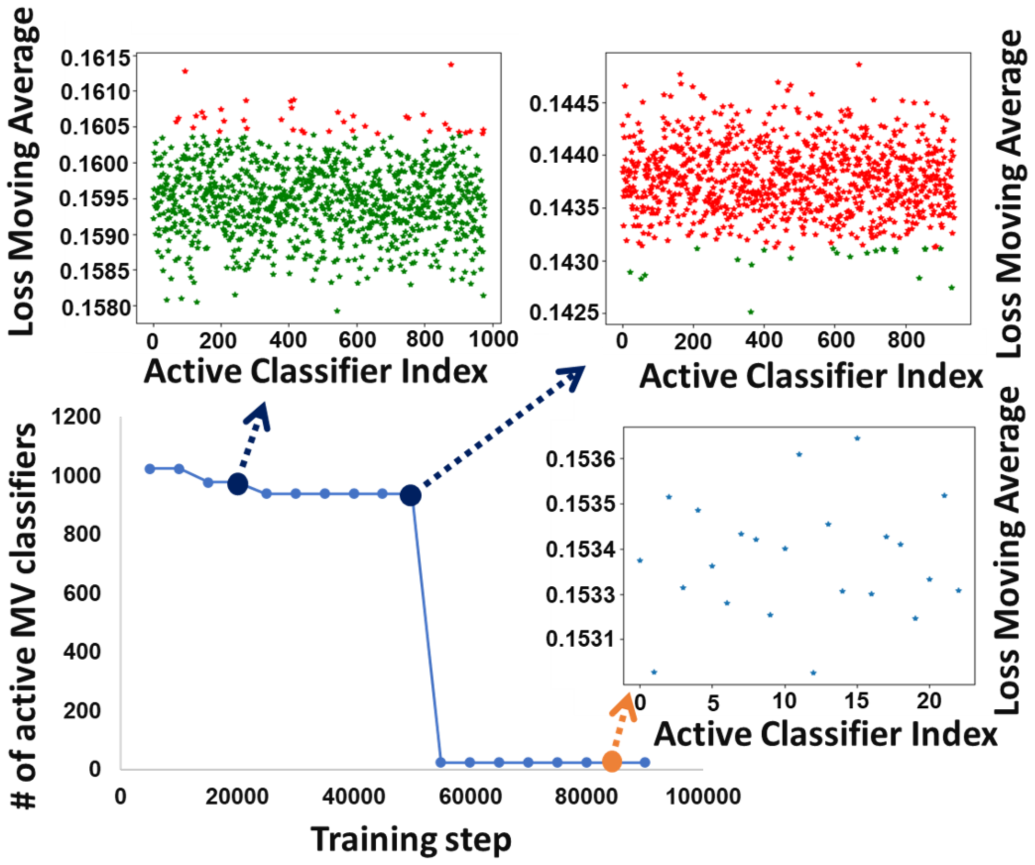


Figure 2: The number of active multiverse classifiers, per training step, for the MML model trained on QNLI. The MeanShift algorithm detects multiple clusters, for three times during training. The two upper plots present the selection of the top-performing heads (green stars), and the elimination of the heads showing lower-performance (red stars). The Y values are the moving average calculated on the multiverse heads' loss function. Our MML-QNLI model reaches local minima at step 85K, for which 23 heads were activated. The bottom right plot shows the moving average values of the activated 23 multiverse heads, by the same training step.

The BERT-Drop model is a BERT baseline trained with a dropout rate of 30% (instead of 10% as used in a regular BERT). Compared to BERT, BERT-Drop provides similar results on the test set, while showing some improvement on the development set. When comparing BERT-Drop to MML, MML results with a higher average performance on both the test set and development set, where in some dataests, such as RTE, MML yields an improvement of +3.6% on the development set. On the test set, MML surpasses BERT-Drop on seven out of eight datasets. Specifically, in some datasets, such as MRPC and CoLA, MML outperforms BERT-Drop in 1.4 and 2.5 accuracy points, respectively.

The BERT-Ensemble employs BERT with an ensemble of 1024 parallel classifiers. It optimizes all classifiers during training and infers predictions by calculating the mean over all classifiers' logits. As shown in the table, BERT-Ensemble yields similar performance to BERT.

Fig. 2 presents the amount of active multiverse heads, when applying MML on QNLI dataset. During the training of the MML-QNLI model, the MeanShift algorithm detected multiple clusters at three times[1], through the entire training. Each time, the model eliminated the lower-performance subsets, and kept the top-performing multiverse classifiers as the active set of classifiers. The model achieved best performance on the development set at training step 85K. At this step, the MML-QNLI model utilized 23 active multiverse heads. The plots in the figure present the cumulative loss of each multiverse head, sorted through the X axis according to the indices of the active heads. The red stars are asso-

[1] The elimination is being invoked every time the MeanShift algorithm detects multiple clusters. Specifically, for MML-QNLI experiment, multiple clusters appeared three times during the training process.

ciated with the classifiers heads that were eliminated, and the green stars are the heads that were selected as the top-performing subset.

4.2 Cross dataset evaluations

In the cross-task evaluations, we use the finetuned MML models from Sec. 4.1. For each model trained on a dataset from the two first categories above (NLI and similarity/paraphrasing), we evaluate the model on all datasets from the same category. Train and development set performance are reported to give a clear view on the cross-task generalization ability, and also to exhibit the level of overfitting, when evaluating on the same dataset that the model was trained on.

In order to conduct a clean comparison, we finetune BERT, MML, and all other methods with the same hyperparmeters, employing 10/30/100 epochs for the relative large/medium/small datasets, a batch size of 32, and a learning rate of 2e-5 (BERT obtains in all cases performance that at least matches the one published in (Devlin et al., 2018)). Our code can be found at `https://github.com/ItzikMalkiel/MML`.

In cross evaluation experiments, there is no training on the target training set, so both the target training set and validation sets can be used for evaluation. Overall, there are 36 cross-task experiments. In the first category, there are four datasets, so three cross-task experiments for each. Taking into account the two splits, this amounts to 24 experiments. Similarly, in the second category, where there are three datasets, there are 12 experiments. Fig. 3 is a Dolan Moré plot comparing the performance of BERT to the five finetuning variants explored (MML, MV-5, MV-1024, BERT-Drop, Bert-Ensable) for all 36 experiments. These plots show for each method the ratio of experiments for which it has a performance level that is at least as high a constant times the best result. As can be seen, regularization helps cross-task generality. However, none of the baseline methods is as effective as MML. In the supplementary appendix we provide the full data, and, for brevity, below we focus on comparing MML with BERT.

Inference tasks Since MNLI and SNLI are multiclass classification tasks with 3 classes, we collapse the labels "neutral" and "contradication" into one label ("non entailment"). This modification, applied only during inference, allows us to evaluate MNLI and SNLI models on RTE and

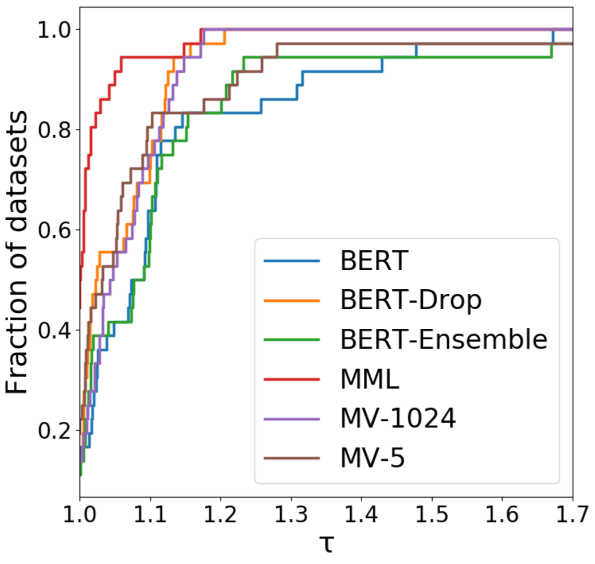


Figure 3: A Dolan-Moré profile, based on the results obtained across all cross-task experiments. The x-axis is the threshold τ. The y-axis depicts, for a given fine-tuning method, the ratio of datasets in which the obtained error is less than the threshold τ times the minimal error obtained by any of the six methods.

QNLI, and vice versa.

The results are reported in Tab. 2. As can be seen, MML exhibits a significantly improved transferability compared to BERT. Each row in the table represents an MML or BERT model trained on a single dataset associated by its name. All models are evaluated on all four datasets. In the last column, we report the relative average improvement obtained by MML, calculated by the performance ratio between MML and BERT across all three holdout datasets. For example, for RTE, our MML-RTE model yields 9.9% relative average performance on the train set of MNLI, QNLI and SNLI, and a 9.5% average improvement on the development set of these datasets.

Similarity and paraphrasing Since STS-B is a regression task benchmark, while MRPC and QQP address a binary classification task, to support cross dataset evaluations, we adapt STS-B to form a binary classification task. The adaptation is being done by collapsing the labels in the range 1-2 (4-5) to the value of 0 (1) and omitting all samples associated with label values between 2 and 4. The binary STS-B version has ~3.5K samples.

As can be seen in Tab. 3, MML yields better performance on the cross evaluations for the similarity and paraphrase datasets. Similar to Tab. 2, each row represent a single model trained on a single dataset. We evaluate all models on all three

Model	RTE	MNLI	QNLI	SNLI	MML Improvement
BERT-RTE	96.06/70.39	69.42/69.17	52.46/52.84	68.02/69.87	-
MML-RTE	99.39/**80.14**	79.24/78.42	50.86/51.30	80.85/82.53	+9.98%/+9.51%
BERT-MNLI	79.15/76.89	99.59/86.58	49.88/51.05	81.65/83.65	-
MML-MNLI	79.35/78.70	99.74/**87.18**	49.64/50.22	82.37/83.94	+0.21% /+0.35%
BERT-QNLI	53.37/48.73	59.76/59.89	99.99/**94.01**	59.33/60.03	-
MML-QNLI	53.41/53.79	64.93/63.85	95.75/92.86	62.13/63.58	+4.48%/+7.63%
BERT-SNLI	71.28/70.03	75.82/75.79	49.62/50.72	99.74/91.08	-
MML-SNLI	71.88/69.31	75.79/76.37	49.32/50.32	99.30/**91.38**	+0.07%/-0.36%

Table 2: Cross dataset evaluation for Language Inference tasks. Train/development accuracy are reported separately for each dataset. Each model (a row in the table) was trained on a single dataset denoted by its name, and was evaluated on the train/development sets of all four datasets. The last columns indicates the relative average improvement obtained by MML compared to BERT, and averaged across the three hold-out datasets (i.e. excluding the diagonal). SNLI is the only dataset for which MML does not improve cross dataset performance on average, perhaps since it is largest dataset with 570k samples. See supplementary appendix for farther comparison with MV-5, MV-1024 and BERT-Drop.

Model	QQP	MRPC	STS-B*	MML Improvement
BERT-QQP	99.73/91.57	66.90/68.85	88.34/90.12	-
MML-QQP	99.74/**91.68**	67.77/68.87	89.11/90.55	+1.08%/+0.25%
BERT-MRPC	65.28/65.18	99.37/87.25	82.53/88.58	-
MML-MRPC	68.37/68.15	99.23/**88.97**	86.12/91.32	+4.54%/+3.82%
BERT-STS-B*	73.13/73.11	75.59/75.49	100.0/95.49	-
MML-STS-B*	74.13/74.40	75.51/77.94	99.85/**96.70**	+0.63%/+2.50%

Table 3: Cross dataset evaluation for similarity and paraphrasing tasks. STS-B* is the modified version of STS-B that forms a binary classification dataset (instead of regression). STS-B* models were trained as binary classifiers on STS-B data. Accuracy scores are reported through all evaluations. The last column presents the relative cross dataset improvement obtained by MML, compared to BERT.

datasets, and report the average relative improvement obtained by MML calculated on the two holdout datasets. We have found MML to produce improved performance for all models, for example, MML-MRPC yields a ~+3.5% average improvement calculated on both train and development sets across STS-B and QQP.

5 Discussion

The results in both Tab. 2 and 3, reveal a significant gap in performance for all models when evaluated on holdout datasets, although the holdout datasets share the same or similar task each model was trained for. For example, both MML-MRPC and BERT-MRPC models yield a ~20% degradation in absolute accuracy on the QQP dataset. However, compared to BERT, our MML method produces significantly better performance on the cross evaluations. Specifically, when evaluated on QQP, MML-MRPC outperforms BERT-MRPC by a relative improvement of ~4.6%, for both development and train set.

We do not observe a direct link between the improvement obtained on the same dataset evaluation to that obtained in the cross dataset one. For example, our MML-QNLI model was able to outperform BERT-QNLI in the cross dataset evaluation, although, compared to our BERT-QNLI reproduction, MML-QNLI exhibits a somewhat degraded performance on QNLI's development set (see the third section in Tab.2) and test set (as published in (Devlin et al., 2018)). We attribute this to the ability of MML to encourage the model to produce more transferable coding vectors.

Computational overhead During training, the MML multiverse heads imply a maximal addition of 1024 operations of matrix multiplications and gradient derivations, each in the size of $1024 \times c$.

During the backward steps, the multiple gradients are averaged almost immediately, right before propagating them back through the last encoder layer of BERT. Interestingly, in many cases, the number of active heads shrinks in the early stages of the training. During inference, the additional calculations are solely the matrix multiplications, for which the average number of active heads, across our experiments, is 381. This average number of active heads translates to an increase of ~0.2% to the parameter count of BERT_Large.

6 Summary

We introduce the MML method for fine-tuning language models. MML utilizes a large set of parallel multiverse heads and eliminates the relatively weaker heads during training. We demonstrate the effectiveness of MML on nine common NLP datasets, by applying inter- and intra-datasets evaluation, where it is shown to outperform the originally introduced BERT fine-tuning procedure. The results shade light on the role of regularization in improving cross task generalization, and show the advantage of MML over alternative regularization methods.

Acknowledgment

This project has received funding from the European Research Council (ERC) under the European Union Horizon 2020 research and innovation programme (grant ERC CoG 725974). The contribution of the first author is part of a PhD thesis research conducted at Tel Aviv University.

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs.

Dorin Comaniciu and Peter Meer. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 603–619.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

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.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Etai Littwin and Lior Wolf. 2016. The multiverse loss for robust transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3957–3966.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *openai blog*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew 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.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru

Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Supplementary Appendices[2]

A The datasets (more details)

We adopt eight datasets from the GLUE benchmark (Wang et al., 2018), and one extra dataset supporting the task of Natural Language Inference (NLI). The datasets can be arranged by categories as follows.

Inference tasks This category contains three datasets from the GLUE benchmark (Wang et al., 2018), along with an external dataset named SNLI (Bowman et al., 2015) that shares the same task. **RTE** The Recognizing Textual Entailment dataset (Bentivogli et al., 2009) is composed of sentence pairs associated with a binary classification task for entailment/non entailment relation between the sentences. **MNLI** Multi-Genre Natural Language Inference Corpus (Williams et al., 2017) is also an entailment dataset comprised of sentence pairs. The task is multiclass classification for predicting contradiction, neutral or entailment relation between the sentence pairs. **SNLI** The Stanford Natural Language Inference dataset (Bowman et al., 2015) is similar to MNLI, but contains data gathered from different sources. **QNLI** The Question-answering Natural Language Inference dataset (Rajpurkar et al., 2016) contains question-sentence pairs associated with the binary classification task for the entailment/non entailment relation between the question-answer pairs.

[2]Put here for the reader's convenience.

Similarity and paraphrasing Tasks This category contains three datasets. **MRPC** Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a dataset of sentence pairs. The task is to determine whether a pair of sentences are semantically equivalent (binary classification). **QQP** Quora Question Pairs (Chen et al., 2018) is a dataset of question pairs taken from the Quora website. The goal is to determine whether a pair of questions are semantically equivalent (binary classification). **STS-B** Semantic Textual Similarity Benchmark (Cer et al., 2017) is a dataset composed of sentence pairs, each annotated with a score between 1 and 5, indicating the semantic similarity level of both sentences. The task is to predict these scores (regression).

Misc. datasets The two datasets in this category are not used for the cross dataset evaluation, due to the lack of commonality between their tasks. **CoLA** The Corpus of Linguistic Acceptability dataset (Warstadt et al., 2018) consists of examples of expert English sentence acceptability judgments. Each sample is annotated by whether it is a grammatically sentence of English (binary classification). **SST-2** The Stanford Sentiment Treebank (Socher et al., 2013) is a dataset composed of sentences assigned with a human annotations of their sentiment. The task is to determine whether the sentiment of each sentence is positive or negative (binary classification).

B Cross-task generalization results for all baselines

For the sake of completeness, we report cross dataset evaluations, for all six models described in the main text. Specifically, we compare MML, BERT, MV-5, MV-1024, BERT-Drop and BERT-Ensemble on cross dataset evaluations for inference datasets, and similarity and paraphrase datasets.

Tab. 4 presents the performance of the above six models evaluated on the cross dataset evaluation for the NLI category. Each row represents a different model, trained on a single dataset associated by its name. Each column corresponds to an evaluation on four datasets in this category. The last column presents the improvement obtained by each model, compared to BERT, and calculated across the three hold-out datasets. Accuracy is reported for all datasets on both the train and development sets, in the format train accuracy/development accuracy. Note that no training is done on the training set in the case of cross-task evaluation. Therefore, the training set can be seen as an additional dataset to evaluate on.

As can be seen in Tab. 4, MML-RTE outperforms BERT-RTE on all datasets, and results with similar performance compared to MV-5-RTE, MV-1024-RTE and BERT-Drop-RTE. For the models trained with MNLI dataset, we found that the four models MML-MNLI, MV-5-MNLI, MV-1024-MNLI and BERT-Drop-MNLI, yield similar results, where all produce a slight improvement compared to BERT. For QNLI, MML yields a significant improvement compared to all models. Specifically, MML-QNLI yields +4.48%/7.63% relative improvement compared to BERT, averaged on the train and development set of the three hold-out dataests. BERT-Drop-QNLI yields significantly decreased performance of -1.93%/-2.89% compared to BERT, MV-5 yields improved performance of +1.0%/+2.73%, and MV-1024 results with a significantly decreased performance of -2.64% on the train sets, and somehow similar performance to BERT on the development set.

For SNLI, MML and BERT-Drop yield similar performance compared to BERT on the training sets, where MML shows slightly better performance. On the development set, both MML and Bert-Drop present slightly decreased performance. Mv-5 and Mv-1024 exhibit significantly decreased performance on both train and development sets.

In Tab.5, we present the performance of all six fine-tune variants, evaluated on the cross similarity and paraphrasing datasets. For QQP-based models, MML and MV-5 were able to improve the cross evaluation performance compared to BERT, measured on the two hold-out datasets. Both MML-QQP and MV-5-QQP yield more than one percentage of relative improvement on the train sets, and less than one percentage of improvement on the developments sets, where MV-5-QQP presents slightly better results. On the other hand, BERT-Drop-QQP and MV-1024-QQP yield significantly decreased performance compared to BERT, each results with 1.76%-3.22% of decreased performance on the train/development sets.

For MRPC, MML outperforms all models, showing a significantly improvement of +4.54%/3.82% on the train/development sets, when compared to BERT. The other models where able to improve BERT in +2.69% up to +3.89% on the train and +1.66% up to +2.95% on the development sets, which are significantly inferior to MML-MRPC. For STS-B, we see that MML, MV-5 and BERT-Drop were able to yield similar improvement compared to BERT, where MV-1024 yields decreased performance on the train sets, and similar performance on the development set.

All in all, MML gained the highest improvement, accumulated across all evaluations. Specifically, when averaging the cross evaluation results over the train sets, MML shows an average improvement of +3.0%, BERT-Drop shows average improvement of +1.64%, MV-5 yields +2.27% and MV-1024 yields +0.46%.

Model	**RTE**	**MNLI**	**QNLI**	**SNLI**	**Improvement compared to BERT**
BERT-RTE	96.06/70.39	69.42/69.17	52.46/52.84	68.02/69.87	-
MML-RTE	99.39/**80.14**	79.24/78.42	50.86/51.30	80.85/82.53	+9.98%/+9.51%
MV-1024-RTE	99.39/**80.14**	79.24/78.42	50.86/51.30	80.85/82.53	+9.98%/+9.51%
MV-5-RTE	100.0/75.45	74.50/73.73	58.22/58.72	75.48/77.15	+9.75% /+9.36%
BERT-D-RTE	100.0/76.53	79.31/78.29	52.55/52.80	79.53/81.10	+10.4% /+9.71%
BERT-Ensemble-RTE	96.06/70.39	70.31/70.02	51.25/51.89	69.54/69.97	-1.20%/+0.34%
BERT-MNLI	79.15/76.89	99.59/86.58	49.88/51.05	81.65/83.65	-
MML-MNLI	79.35/78.70	99.74/**87.18**	49.64/50.22	82.37/83.94	+0.21% /+0.35%
MV-1024-MNLI	78.91/78.70	98.71/86.24	49.93/50.48	82.33/83.60	+0.20% /+0.39%
MV-5-MNLI	78.83/77.61	98.11/87.00	49.91/50.46	82.51/83.95	+0.23% /+0.04%
BERT-D-MNLI	79.40/78.70	99.90/87.04	49.75/50.22	82.32/83.71	+0.29% /+0.26%
BERT-Ensemble-MNLI	78.68/75.58	99.59/86.58	50.21/52.21	82.51/82.35	+1.54%/+0.54%
BERT-QNLI	53.37/48.73	59.76/59.89	99.99/**94.01**	59.33/60.03	-
MML-QNLI	53.41/53.79	64.93/63.85	95.75/92.86	62.13/63.58	+4.48%/+7.63%
MV-1024-QNLI	51.04/51.26	60.28/58.6	99.49/92.24	58.68/58.78	-2.64%/+0.31%
MV-5-QNLI	51.12/50.90	61.58/60.36	97.95/92.18	61.81/61.86	+1.00%/+2.73%
BERT-D-QNLI	49.91/45.84	60.48/59.33	99.99/93.17	59.22/59.14	-1.93%/-2.89%
BERT-Ensemble-QNLI	54.52/49.84	58.84/59.25	98.95/93.01	60.35/61.87	-1.75%/-1.02%
BERT-SNLI	71.28/70.03	75.82/75.79	49.62/50.72	99.74/91.08	-
MML-SNLI	71.88/69.31	75.79/76.37	49.32/50.32	99.30/**91.38**	+0.07%/-0.26%
MV-1024-SNLI	68.87/64.98	73.41/74.09	48.89/49.33	94.54/91.16	-2.67% /-4.06%
MV-5-SNLI	71.24/67.87	75.38/75.00	49.42/50.35	95.63/91.18	-0.35% /-1.62%
BERT-D-SNLI	71.36/70.84	76.16/76.38	49.32/49.35	95.65/91.45	-1.35% /-1.24%
BERT-Ensemble-SNLI	72.52/71.89	74.75/74.85	50.54/51.02	99.57/91.2	-1.23%/-1.10%
MML impr.	+0.38%/+3.90%	+7.58%/+6.91%	-1.37%/-1.77%	+8.15%/+8.12%	
MV-1024 impr.	-2.68%/+0.11%	+3.94%/+2.99%	-1.47%/-2.25%	+6.19%/+5.32%	
MV-5 impr.	-1.55%/+0.76%	+3.26%/+2.11%	+3.54%/3.08%	+5.40%/-0.94%	
BERT-D impr.	-2.01%/-0.80%	+5.29%/+4.34%	-0.23%/-1.46%	+5.85%/+4.88%	
BERT-Ensemble impr.	-1.62%/-1.40%	+1.64%/+0.51%	-0.52%/-0.54%	-1.66%/-0.55%	

Table 4: Cross dataset evaluation for Language Inference tasks for all ablation models presented in the main text (BERT-D stands for BERT-Drop). Train/development accuracy are reported separately for each dataset. Each model (a row in the table) was trained on a single dataset denoted by its name, and was evaluated on the train/development sets of all four datasets. The last column indicates the relative average improvement obtained by each model compared to BERT, and averaged across the three hold-out datasets. The last four rows present the average column-wise improvement each technique yields, which is aggregated from the three models of the same technique. BERT models were reproduced with the same hyperparamters used for MML (all BERT reproductions result with similar or better performance compared to the one published in (Devlin et al., 2018)).

Model	**QQP**	**MRPC**	**STS-B***	**Improvement compared to BERT**
BERT-QQP	99.73/91.57	66.90/68.85	88.34/90.12	-
MML-QQP	99.74/91.68	67.77/68.87	89.11/90.55	+1.08%/+0.25%
MV-1024-QQP	98.78/91.42	65.59/67.40	88.20/90.01	-2.16%/-3.22%
MV-5-QQP	99.40/91.45	68.19/68.60	89.00/91.09	+1.58%/+0.63%
BERT-Drop-QQP	99.57/91.64	64.50/65.68	88.40/89.24	-1.76% /-2.80%
BERT-Ensemble-QQP	99.73/91.42	67.45/69.85	89.41/90.45	-1.51%/-1.18%
BERT-MRPC	65.28/65.18	99.37/87.25	82.53/88.58	-
MML-MRPC	68.37/68.15	99.23/88.97	86.12/91.32	+4.54%/+3.82%
MV-1024-MRPC	67.33/67.44	99.91/89.70	85.19/90.55	+3.18%/+2.76%
MV-5-MRPC	66.69/66.46	99.94/87.99	85.20/89.79	+2.69%/+1.66%
BERT-Drop-MRPC	67.65/67.66	99.97/87.5	85.97/90.45	+3.89% /+2.95%
BERT-Ensemble-MRPC	64.48/64.88	99.56/87.25	81.23/87.98	+1.58%/+1.27%
BERT-STS-B*	73.13/73.11	75.59/75.49	100.0/95.49	-
MML-STS-B*	74.13/74.40	75.51/77.94	99.85/96.70	+0.63%/+2.50%
MV-1024-STS-B*	73.96/74.25	75.59/77.20	99.85/96.48	-2.64%/+0.31%
MV-5-STS-B*	74.93/75.15	75.21/75.98	100.0/97.03	+0.97%/+1.71%
BERT-Drop-STS-B*	74.29/75.45	75.40/76.22	100.0/96.70	+0.66% /+2.08%
BERT-Ensemble-STS-B*	74.45/74.21	76.01/76.45	100.0/96.15	-1.27%/-1.08%
MML impr.	+3.05%/+3.16%	+0.59%/+1.63%	+2.61%/+1.78%	
MV-1024 impr.	+2.13%/+2.51%	+0.97%/+0.07%	+1.53%/+1.05%	
MV-5 impr.	+2.31%/+2.37%	+0.71%/+0.14%	+1.99%/+1.22%	
BERT-Drop impr.	+2.60%/+3.50	-1.91%/-1.81%	+2.11%/+0.56%	
BERT-Ensemble impr.	-0.31%/-1.02%	-1.42%/-1.28%	+0.21%/+0.19%	

Table 5: Cross dataset evaluation for similarity and paraphrase tasks. STS-B* is the modified version of STS-B that forms a binary classification dataset (instead of regression). STS-B* models were trained as binary classifiers on STS-B data. Accuracy scores are reported through all evaluations. The last column presents the relative cross dataset improvement obtained by each model compared to BERT, and averaged across the two hold-out datasets. The last four rows present the average column-wise improvement each technique yields, which is computed from the two models of the same technique.