# Beyond Reptile: Meta-Learned Dot-Product Maximization between Gradients for Improved Single-Task Regularization

**Akhil Kedia** and **Sai Chetan Chinthakindi** and **Wonho Ryu**
Samsung Research, Seoul, South Korea
[akhil.kedia, sai.chetan, wonho.ryu]@samsung.com

## Abstract

Meta-learning algorithms such as MAML, Reptile, and FOMAML have led to improved performance of several neural models. The primary difference between standard gradient descent and these meta-learning approaches is that they contain as a small component the gradient for maximizing dot-product between gradients of batches, leading to improved generalization. Previous work has shown that aligned gradients are related to generalization, and have also used the Reptile algorithm in a single-task setting to improve generalization. Inspired by these approaches for a single task setting, this paper proposes to use the finite differences first-order algorithm to calculate this gradient from dot-product of gradients, allowing explicit control on the weightage of this component relative to standard gradients. We use this gradient as a regularization technique, leading to more aligned gradients between different batches. By using the finite differences approximation, our approach does not suffer from $\mathcal{O}(n^2)$ memory usage of naively calculating the Hessian and can be easily applied to large models with large batch sizes. Our approach achieves state-of-the-art performance on the Gigaword dataset, and shows performance improvements on several datasets such as SQuAD-v2.0, Quasar-T, NewsQA and all the SuperGLUE datasets, with a range of models such as BERT, RoBERTa and ELECTRA. Our method also outperforms previous approaches of Reptile and FOMAML when used as a regularization technique, in both single and multi-task settings. Our method is model agnostic, and introduces no extra trainable weights.

## 1 Introduction

Meta-learning algorithms such as MAML (Finn et al., 2017), FOMAML, and Reptile (Nichol et al., 2018), which modify gradient descent by effectively differentiating through it, have lead to performance improvements on several datasets such as MiniImageNet (Vinyals et al., 2016), Omniglot (Lake et al., 2011) and Java Github Corpus (Allamanis and Sutton, 2013). When used in a single-task setting, the only significant difference between these algorithms and standard SGD is that the meta-gradient from these algorithms also contains as a component the gradient for maximizing the dot-product between gradients for examples/batches, as was theoretically proven in the Reptile paper.

The Reptile algorithm has also been leveraged to improve single-task performance across a range of models and tasks, such as in Kedia and Chinthakindi (2021). Second-order methods for aligned gradients have been explored before in the context of continual learning, such as in Riemer et al. (2019), Lopez-Paz and Ranzato (2017), Chaudhry et al. (2018). Some recent work, such as Fort et al. (2019), Chatterjee (2020), and Yu et al. (2020) have also shown that aligned gradients are related to improved generalization and model performances. We conjecture that aligned gradients in single-task settings will also improve learning across examples, enabling better transfer from one example to another, similar to as often done in continual/multi-task approaches such as Riemer et al. (2019).

However, a naive approach to directly maximize the dot-product of gradients requires a calculation of the Hessian Matrix, which scales as $\mathcal{O}(n^2)$ in memory usage where $n$ is the number of model parameters. This approach also fails to work with gradient accumulation, leading to a hard limit on the batch size, reducing training accuracy. Even though some recent works such as Anil et al. (2020) have tried to make this tractable using large distributed environments, the computation costs are extremely high for any reasonably large model. Approaches like the Hessian-Vector Product (Pearlmutter, 1994) also do not work with gradient accumulation.

Inspired by the above approaches and to fix the

*Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 407–420
November 7–11, 2021. ©2021 Association for Computational Linguistics

aforementioned issues, we propose to explicitly calculate the gradient for maximizing the dot product between batches using a first-order approximation. We use this gradient as a regularizing term, and show that it results in improved performance across a range of models and tasks. Our main contributions are-

- Use the first-order finite differences method (Smith et al., 1985) to explicitly calculate the gradient from the dot-product of gradients.
- Utilize the above gradient to regularize the training of models in single task settings.
- Leads to significant performance improvements across a wide range of tasks and datasets such as SQuAD-v2.0, Quasar-T and all SuperGLUE datasets. Achieves State-of-the-art performance on Gigaword.
- Outperforms previous approaches such as Reptile and FOMAML in single-task as well as multi-task setting.
- Is model agnostic, with no extra trainable weights.
- Improves performance across a range of model sizes and pre-training, such as BERT, ELECTRA, RoBERTa, and for small, base and large models.

## 2 Proposed Method

### 2.1 Background on Reptile and MAML Algorithms

**MAML** The MAML algorithm, initially intended for multi-task few-shot learning, proposed to do $k$ steps of "inner" gradient updates, after which the loss was computed and minimized on the $(k+1)^{th}$ batch, with respect to the original weights before the $k$ inner steps. The gradient from this loss is then used for an "outer" update to the original weights. This requires differentiating through the optimizer, and is a second order method.

**FOMAML** The authors of MAML also proposed FOMAML, which is a first order approximation of MAML bypassing the differentiation through the optimizer. This method also achieves significant improvement compared to vanilla learning algorithms.

**Reptile** The Reptile algorithm is similar to FOMAML, and also does $k$ inner steps of gradient updates. For the outer update, Reptile uses the difference between the original weights and the inner weights as the gradient. The Reptile paper showed that the gradient for all these 3 approaches is similar to vanilla SGD, except for a small component which maximizes dot-product between batches -

$$G_{Reptile} = (k)G_{avg} - (\frac{\alpha}{2}k(k-1))G_{Inner}$$

where $G_{avg}$ is the expected SGD gradient from a batch, $\alpha$ is the inner-step size, and $G_{Inner}$ is the gradient for maximizing dot-product between batches. The gradient is similar for MAML and FOMAML, only differing in the constants. But this approximation is only valid for small $\alpha$, which reduces the ability of $G_{inner}$ to regularize the training. By computing $G_{inner}$ explicitly, we aim to overcome this limitation.

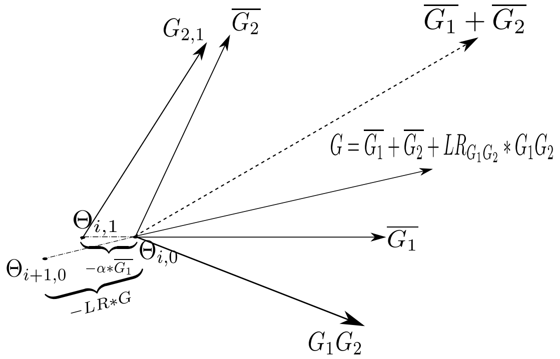
$G_{2,1}$
$\overline{G_2}$
$\overline{G_1} + \overline{G_2}$
$G = \overline{G_1} + \overline{G_2} + LR_{G_1G_2} * G_1G_2$
$\Theta_{i,1}$
$\overline{G_1}$
$\Theta_{i,0}$
$\Theta_{i+1,0}$
$-\alpha * \overline{G_1}$
$-LR * G$
$G_1G_2$

Figure 1: Our proposed algorithm: Calculating the gradient for maximizing dot product using finite-differences approximation, and using it for regularization of standard gradient.

### 2.2 Our Approach: Meta DotProd

Our proposed regularization scheme is inspired by the inner loop of the Reptile algorithm, and uses the finite-differences method to approximate the Hessian-vector product. Algorithm 1 shows how to calculate the gradient for maximizing the dot product of gradients using the finite differences method applied to an SGD optimizer. Essentially, we calculate the gradients $\overline{G_1}$ and $\overline{G_2}$ from batches $b_1$ and $b_2$, and then temporarily update the network parameters with $\alpha * \overline{G_1}$. Then we calculate $G_{2,1}$ with batch $b_2$ again with the new network parameters and use this gradient to calculate the dot product gradient $G_1G_2$.

Once this gradient of dot product is calculated, unlike Reptile and FOMAML, we can explicitly control its relative weight by adjusting the hyperparameter $LR_{G_1G_2}$.

```
Algorithm 1: META DOTPROD
Input: Batches $B = \{b_0, b_1, \dots, b_n\}$
       $LR$ = Learning Rate
       $LR_{G_1G_2}$ = Weight of $G_1.G_2$ grad
       $\theta_{0,0}$ = Network Weights
       $\alpha$ = A small constant
Output: Final fine-tuned $\theta$
for $i \leftarrow 0$ to $\lfloor n/2 \rfloor$ do
    $\overline{G_1} \leftarrow$ grad from $\theta_{i,0}(b_{i*2})$
    $\overline{G_2} \leftarrow$ grad from $\theta_{i,0}(b_{i*2+1})$
    $\theta_{i,1} \leftarrow \theta_{i,0} - \alpha * \overline{G_1}$
    $G_{2,1} \leftarrow$ grad from $\theta_{i,1}(b_{i*2+1})$
    $G_1G_2 \leftarrow ((G_{2,1} - \overline{G_2})/\alpha)$
    $G \leftarrow \overline{G_1} + \overline{G_2} + LR_{G_1G_2} * G_1G_2$
    $\theta_{i+1,0} \leftarrow \theta_{i,0} - LR * G$
return $\theta_{\lfloor n/2 \rfloor+1,0}$
```

The last line of the algorithm is the SGD update, which can be substituted by any other optimizer. In our experiments, we use the original model's default optimizer, which range from SGD to Adam (Kingma and Ba, 2015) to AdaFactor (Shazeer and Stern, 2018).

Gradient accumulation, if any, can be stored in $\overline{G_1}$, $\overline{G_2}$ and $G_{2,1}$ before applying this algorithm. The compute and storage needed for our method scales linearly with model size (approx. 50%), allowing us to apply this to large models such as BERT, with significantly smaller overhead compared to calculating the Hessian. This overhead can be reduced to 10% without significantly impacting our method, as we show in subsection 6.3.

## 3 Theoretical Analysis

In this section, we provide a theoretical analysis of our meta update of the $Meta\ DotProd$ algorithm. We generalize the Taylor expansion approach as used in Nichol et al. (2018), and show how our approach maximizes inner product of gradients between different mini-batches. This approach is essentially the expectation of the finite differences method over stochastic mini-batch sampling for calculating the Hessian-Vector product, but we present it here for clarity.

We consider two input batches $b_0$, $b_1$ at the beginning of $i^{th}$ step. For $j \in \{0, 1\}$ we define -

$$\begin{aligned}
\theta_{i,0} &= \text{network weights before } i^{th} \text{ step},\\
L_{j+1} &= \text{loss function corresponding to } b_j,\\
G_{j+1} &= L'_{j+1}(\theta_{i,j}), && \text{(gradient of } b_j)\\
\overline{G_{j+1}} &= L'_{j+1}(\theta_{i,0}), && \text{(gradient at initial point)}\\
H_{j+1} &= L''_{j+1}(\theta_{i,j}), && \text{(Hessian of } b_j)\\
\overline{H_{j+1}} &= L''_{j+1}(\theta_{i,0}), && \text{(Hessian at initial point)}\\
\alpha &= \text{A small constant,}
\end{aligned}$$

Then, our update rules are -

$$\theta_{i,1} = \theta_{i,0} - \alpha * \overline{G_1}, \quad (1)$$

$$G_{2,1} = G_2(\theta_{i,1}) = G_2(\theta_{i,0} - \alpha * \overline{G_1}) \quad (2)$$

Using the first order Taylor expansion of $G_2$, we get -

$$G_{2,1} = \overline{G_2} - \alpha \overline{G_1}\,\overline{H_2} + O(\alpha^2), \quad (3)$$

$$\frac{(G_{2,1} - \overline{G_2})}{\alpha} = -\overline{G_1}\,\overline{H_2} + O(\alpha), \quad (4)$$

For small $\alpha$, we can ignore the terms involving $O(\alpha^2)$ in (3). This term becomes $O(\alpha)$ in (4), but it is still $\alpha = 1e^{-7}$ times smaller than $\overline{G_1}\,\overline{H_2}$ and hence can be safely ignored.

Under the expectation of stochastic mini-batch sampling, $E[\overline{G_1}\,\overline{H_2}] = E[\overline{G_2}\,\overline{H_1}]$, and the above equation (4) becomes -

$$\begin{aligned}
E[-\frac{(G_{2,1} - \overline{G_2})}{\alpha}] &= E[\overline{G_1}\,\overline{H_2}], && (5)\\
&= \frac{1}{2} * E[\overline{G_1}\,\overline{H_2} + \overline{G_2}\,\overline{H_1}],\\
&= \frac{1}{2} * E[\frac{\partial(\overline{G_1}.\overline{G_2})}{\partial\theta}], && (6)
\end{aligned}$$

giving exactly the gradient for maximizing the dot product between the gradients.

Note that the above approximation relies on $\alpha$ being a small enough value - If $\alpha$ is too large, the approximation breaks down, and the performance improvement decreases. This is particularly relevant as the relative weight of $G_1.G_2$ component in the Reptile and FOMAML algorithms is directly proportional to this $\alpha$ - limiting the ability to adjust the importance of $G_1.G_2$, and hence limiting performance, as we will show in section 5.

| Corpus | Task | \|Train\| | \|Dev\| |
|---|---|---|---|
| BoolQ | QA | 9.4K | 3.2K |
| CB | NLI | 250 | 57 |
| COPA | QA | 400 | 100 |
| MultiRC | QA | 5.1K | 953 |
| ReCoRD | QA | 101K | 10K |
| RTE | NLI | 2.5K | 278 |
| WiC | WSD | 6K | 638 |

Table 1: Description of datasets in SuperGLUE.

| Corpus | Task | \|Train\| | \|Test\| |
|---|---|---|---|
| SQuAD v2.0 | MRC | 130K | 12K |
| NewsQA | MRC | 97K | 5.4K |
| Quasar-T Long | MRC | 26.3K | 2.1K |
| Quasar-T Short | MRC | 25.4K | 2K |
| Gigaword | Summ. | 3.8M | 1.9K |
| Omniglot | Img. | 24K | 8.4K |
| Mini-Imagenet | Img. | 38K | 12K |

Table 2: Description of NLP and image datasets. For SQuAD and Quasar-T, column 4 refers to validation.

## 4 Experimental Setup

### 4.1 Benchmark Datasets

We describe the dataset size and tasks for each of our datasets in Table 1 and Table 2, and give a short description below.

**SuperGLUE** A popular NLP benchmark, which attempts to test various capabilities of language understanding. It itself consists of 8 datasets - Boolean Questions (Clark et al., 2019), Commitment Bank (De Marneffe et al., 2019), Choice of Possible Alternative (Gordon et al., 2012), Multi-Sentence Reading Comprehension (Khashabi et al., 2018), Reading Comprehension with Commonsense Reasoning (Zhang et al., 2018), Recognizing Textual Entailment (a combination of datasets from Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009; Poliak et al., 2018), Word-in-Context (Pilehvar and Camacho-Collados, 2019) and Winograd Schema Challenge (Levesque, 2011). We omit WSC in our results as both BERT and RoBERTa gave trivial degenerate results.

**SQuAD v2.0** A popular span-style QA dataset, consisting of passages from Wikipedia, with questions and corresponding answer spans and unanswerable questions.

| Model | Params | Speed |
|---|---|---|
| Conv-4 | 351K | ~1K |
| Electra-small | 16M | 125 |
| BERT-base-uncased | 110M | 45 |
| BERT-large-uncased | 340M | 20 |
| RoBERTa-large | 340M | 20 |
| Pegasus-large | 568M | 10 |

Table 3: Models, number of network parameters, and training speeds in examples/second on a V100 GPU.

**Gigaword** Gigaword (See et al., 2017) is an English summarization dataset with single-line input documents from news sources, and the task is to generate headlines.

**Quasar-T** An MRC retrieval dataset from Dhingra et al. (2017), it consists of cloze-style queries constructed from definitions on the website Stack Overflow. It is split into queries with smaller context documents (Quasar-T Short) and with longer context (Quasar-T Long). We only use the subset of the dataset in which the answer is an exact span.

**Omniglot** A dataset containing 20 hand-drawn samples of characters from 50 different alphabets, similar to the popular MNIST (Deng, 2012) dataset. Similar to Finn et al. (2017), we use the first 1200 classes as train and the others as test.

**Mini-Imagenet** A dataset containing 100 random classes from the ImageNet dataset (Deng et al., 2009), resized to 84x84 images, each class having 600 examples.

### 4.2 Models

We describe the model size and speeds of all our models in Table 3 and give a short description below.

**BERT** BERT (Devlin et al., 2019) is a transformer (Vaswani et al., 2017) model, and its derivatives and improvements are the backbone of most state-of-the-art models in NLP. We use the BERT-large-cased official implementation from Jiant (Wang et al., 2019) for SuperGLUE, and the official implementation of BERT-base-uncased and BERT-large-uncased from Rajpurkar et al. (2018) for SQuAD, and re-use the same for QUASAR-T.

**RoBERTa** Roberta is a model with the same architecture as BERT, but the pre-training objectives and parameters are selected more carefully, and

| Corpus | Metric | BERT | +Reptile | +DotProd | RoBERTa | +Reptile | +DotProd |
|---|---|---|---|---|---|---|---|
| BoolQ | Acc | 77.5 ± 0.1 | 77.0 ± 0.2 | **80.0** ± 0.1 | 87.3 ± 0.1 | 87.3 ± 0.1 | 87.3 ± 0.1 |
| CB | F1 | 93.7 ± 0.1 | 93.0 ± 1.1 | **94.2** ± 0.1 | 87.7 ± 4.5 | 91.0 ± 3.2 | **97.5** ± 1.0 |
|  | Acc | 93.8 ± 0.5 | 93.5 ± 0.7 | **95.2** ± 0.6 | 92.6 ± 1.8 | 94.0 ± 1.5 | **97.7** ± 0.8 |
| COPA | Acc | 70.7 ± 1.3 | 70.8 ± 1.4 | **73.4** ± 1.4 | 75.5* | Diverged* | **81.0***|
| MultiRC | F1a | 70.3 ± 0.2 | 70.0 ± 2 | **70.8** ± 0.2 | 78.2 ± 0.4 | 78.2 ± 0.4 | **79.0** ± 0.1 |
|  | EM | 25.9 ± 0.3 | 24.7 ± 0.5 | **27.1** ± 0.4 | 44.0 ± 1.0 | 44.0 ± 1.0 | **45.5** ± 0.3 |
| ReCoRD | F1 | 72.0 ± 0.2 | 70.5 ± 0.5 | **72.5** ± 0.2 | 88.0 ± 0.1 | 87.9 ± 0.1 | 87.9 ± 0.1 |
|  | EM | 71.1 ± 0.2 | 69.8 ± 0.5 | **71.7** ± 0.2 | 88.5 ± 0.1 | 87.3 ± 0.1 | 87.4 ± 0.1 |
| RTE | Acc | 73.4 ± 0.3 | 73.8 ± 0.5 | **74.1** ± 0.4 | 82.7 ± 1.5 | 78.3 ± 5.3 | **85.4** ± 0.4 |
| WiC | Acc | 73.9 ± 0.2 | 73.9 ± 0.2 | 73.9 ± 0.1 | 72.6 ± 0.2 | 72.9 ± 0.3 | 72.9 ± 0.3 |

Table 4: Results on SuperGLUE datasets dev sets, with BERT-Large and RoBERTa-Large models, with BERT hyper-parameters. * 7/8 runs of baseline RoBERTa did not converge for COPA, so we report the (only) best score.

with larger pre-training data. We use the official checkpoints and hyper-parameters from Liu et al. (2019) for SQuAD, and re-use the same for Quasar. As the hyper-parameters for SuperGLUE for Roberta are not available, we re-use the official BERT hyper-parameters.

**ELECTRA** Electra (Clark et al., 2020) pretrains a BERT-like transformer model to discriminate between real and fake input tokens generated by another smaller network. Models deriving from ELECTRA achieve state-of-the-art performance on a range of NLU tasks. We use the ELECTRA-small official implementation for SQuAD.

**Pegasus** A state-of-the-art model for summarization tasks, Pegasus model has the standard base architecture of encoder-decoder transformer, but is pre-trained at the task of generating missing sentences. We use the official model and parameters (Zhang et al., 2020) for Pegasus for Gigaword.

**Conv-4** A convolution network, with 4 blocks of conv2d, batch normalization and relu activation, followed by a dense layer with heads for classification. We use the official model from Nichol et al. (2018) for all our experiments on Omniglot and Mini-ImageNet.

### 4.3 Implementation Details

We train each corresponding model 8 times on each dataset's training set (5 on SQuAD) and report the mean and standard error of these scores. We use one Nvidia V100 for all our experiments (8 for gigaword). As our algorithm is first order, it incurs a linear performance overhead compared to the original model. All experiments run in less than a day, except for Gigaword, MultiRC, ReCoRD and Omniglot-20-way, which run in a few days. As the test sets are hidden for SuperGLUE and SQuAD, we provide results on the dev set instead. We provide dev set results on Quasar-T as well as we use only the subset mentioned above. We only evaluate once on the Gigaword test set, and hence no standard error is provided.

**Hyper-parameters** Details of all default/official model hyper-parameters for each model/dataset, can be found in their source codes, whose links are available in the supplemental material. Wherever official hyper-parameters are not available, we have re-used hyper-parameters from other similar models/datasets, as described in subsection 4.2.

Except in the ablation study for $LR_{G_1G_2}$, we use a fixed value of $\alpha$ as $1e^{-7}$ and $LR_{G_1G_2}$ as 0.1 for all our experiments. $\alpha$ was chosen as this value as it has to be small for the first order approximation to hold. $LR_{G_1G_2}$ was chosen as $10\%$ of the standard gradient so as to not overshadow the standard gradient for the task, while still providing enough gradient to maximize dot product. We keep $k$ for Reptile and FOMAML as 4.

## 5 Results

### 5.1 Results on SuperGLUE datasets

As shown in Table 4, our method consistently improves the performance of both BERT and RoBERTa models on all SuperGLUE datasets. For the BERT model, we show performance gains of 2.5, 1.5, 2.7 and 0.7 in accuracy on BoolQ, CB, COPA and RTE, and 1.2, 0.6 in EM on MultiRC and ReCoRD. With RoBERTa model, we also observe significant performance improvement of 5 and 3 in accuracy on CB and RTE, and 1.5 in EM

| Corpus | Metric | Baseline | +Reptile | +DotProd | RoBERTa | +Baseline | +DotProd |
|---|---|---|---|---|---|---|---|
| SQuAD | F1 | 80.6 ± 0.2 | 81.0 ± 0.2 | **81.9** ± 0.2 | 88.6 ± 0.1 | 88.7 ± 0.1 | **88.9** ± 0.1 |
| | EM | 77.7 ± 0.2 | 78.0 ± 0.1 | **78.9** ± 0.1 | 85.3 ± 0.1 | 85.5 ± 0.1 | **85.7** ± 0.1 |

Table 5: Results on the SQuAD v2.0 with BERT and RoBERTa models respectively.

| Metric | Pegasus | +Reptile | +DotProd |
|---|---|---|---|
| R-1 | 39.7 | 39.8 | **40.6** |
| R-2 | 20.5 | 20.5 | **21.0** |
| R-L | 36.9 | 37.0 | **37.0** |

Table 6: Results on Gigaword test set with PEGASUS. R-1, R-2 and R-L refer to ROUGE-1, ROUGE-2 and ROUGE-L respectively.

| Dataset | F1 | EM |
|---|---|---|
| Quasar-T Long | 75.3 ± 0.2 | 71.7 ± 0.2 |
| +DotProd | **76.1** ± 0.1 | **72.7** ± 0.1 |
| Quasar-T Short | 81.9 ± 0.2 | 78.1 ± 0.2 |
| +DotProd | **82.6** ± 0.1 | **78.9** ± 0.1 |
| NewsQA | 62.6 ± 0.1 | 52.6 ± 0.2 |
| +DotProd | **63.1** ± 0.1 | **52.9** ± 0.1 |

Table 7: Results on the QUASAR-T Long with BERT, Quasar-T short with RoBERTa, and NewsQA with BERT models respectively.

on MultiRC, with minor improvements on other datasets.

### 5.2 Results on other datasets

As shown in Table 5, our method shows performance gains of 1.3 in F1, and 1.2 in EM on SQuAD with the BERT model. Our approach also achieves state-of-the-art performance on the Gigaword dataset, as shown in Table 6. Our approach when applied on the baseline PEGASUS model results in improvement of 0.9 in ROUGE-1, 0.5 in ROUGE-2 and 0.1 in ROUGE-L metrics. We also show performance improvements on Quasar-T dataset, of 0.8 in F1, and 1.0 in EM on the Quasar-T (long) dataset compared to the baseline model of BERT, and of 0.7 in F1, and 0.8 in EM on the Quasar-T (short) dataset compared to the baseline model of RoBERTa as we show in Table 7. Our method also improves the score of the BERT model on the NewsQA dataset by 0.5 in F1.

### 5.3 Results on varying model size and pre-training

To demonstrate the effect of varying model size, as well as improving pre-training, in Table 9 we show the results of using DotProd on Electra-small, BERT-base and RoBERTa-large on SQuAD dataset. Our method improves performance across the entire range of models with varying pre-training strategies and sizes. Furthermore, our method is applicable on even larger models such as Pegasus as shown previously, and on even smaller models such as Conv-4, as we will show in the next section.

### 5.4 Comparison to other Meta-Learning Methods - Few-Shot Multi-task Learning

While the focus of our approach is specifically on single-task learning, we also evaluate our method on few-shot multi-task learning on Omniglot and Mini-Imagenet datasets as done in the Reptile and MAML papers, to see the effectiveness of a higher $G_1G_2$ compared to Reptile and MAML. As shown in Table 8, our method shows consistent improvements against Reptile and FOMAML on the Mini-Imagenet dataset, with a performance gain of 2.53 and 2.77 in 1-shot 5-way and 5-shot 5-way classification respectively against Reptile, and an improvement of 1.53 and 2.36 in 1-shot 5-way and 5-shot 5-way classification against FOMAML.

We also observe consistent performance improvements of 2.19, 0.68, 1.81 and 0.87 against the Reptile approach on 1-shot 5-way, 5-shot 5-way, 1-shot 20-way and 5-shot 20-way classifications respectively on Omniglot dataset. When compared against the FOMAML approach our method achieves a performance gain of 0.38 and 0.55 in 5-shot 5-way and 1-shot 20-way classifications respectively on Omniglot dataset. Note that the Reptile scores and our Dot Prod scores are without trasduction, whereas FOMAML reported scores are transductive, which boosts FOMAML scores.

## 6 Ablation Studies

### 6.1 Comparison to other Meta-Learning Methods - Single Task

In Table 10, we compare our method against the Reptile algorithm and FOMAML, on SQuAD-v2.0 dataset with BERT-large model. Note that the dot-product gradient component in Reptile and FOMAML is directly proportional to $\alpha$, but $\alpha$ has to

| Method | Mini-Imagenet | | Omniglot | | | |
|---|---|---|---|---|---|---|
| | 1-shot 5-w | 5-shot 5-w | 1-shot 5-w | 5-shot 5-w | 1-shot 20-w | 5-shot 20-w |
| Reptile | 47.07±0.26 | 62.74±0.37 | 95.39±0.09 | 98.9±0.10 | 88.14±0.15 | 96.65±0.33 |
| FOMAML | 48.07±1.75 | 63.15±0.91 | **98.3**±0.5 | 99.2±0.2 | 89.4±0.5 | **97.9±0.1** |
| Dot Prod | **49.60**±0.18 | **65.51**±0.26 | 97.58±0.08 | **99.58±0.02** | **89.95±0.05** | 97.52±0.05 |

Table 8: Few-shot Multi-task classification comparison of our method. n-W in the heading refers to n-way classification. Note that FOMAML scores are with transduction, which boosts the scores. All scores are official reported scores.

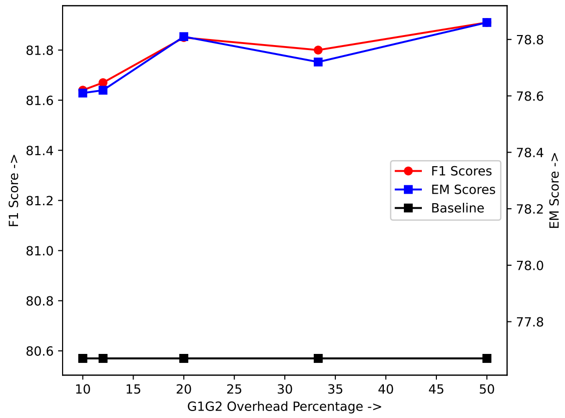

Figure 2: Effect of infrequent regularization and reduced computation overhead on our method, with BERT-large on SQuAD-v2.0 dataset.

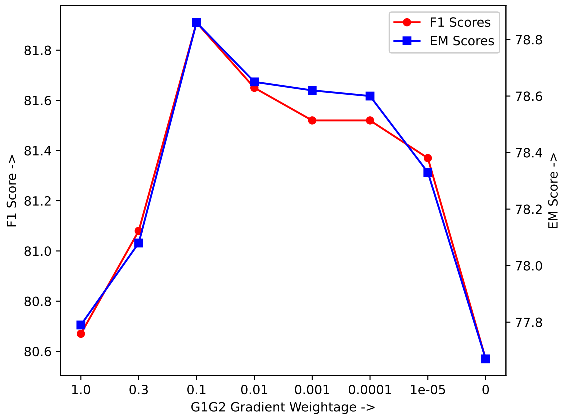

Figure 3: Effect of $LR_{G_1G_2}$ on scores of BERT model on SQuAD-v2.0 dataset. Note that the x-axis is on a logarithmic scale.

| **Method** | **F1** | **EM** |
|---|---|---|
| Electra-small | 71.5 ± 0.1 | 68.9 ± 0.1 |
| +DotProd | **71.9** ± 0.1 | **69.3** ± 0.1 |
| BERT-base | 76.8 ± 0.2 | 73.9 ± 0.2 |
| +DotProd | **77.4** ± 0.1 | **74.4** ± 0.1 |
| RoBERTa-large | 88.6 ± 0.1 | 85.3 ± 0.1 |
| +DotProd | **88.9** ± 0.1 | **85.7** ± 0.1 |

Table 9: Effect of our method on varying the model size and pre-training, from small to large, and from BERT to Electra.

be small for the first-order approximation to hold, representing a direct conflict which limits the performance gains from these methods. Our method does not suffer from this limitation, improving performance.

### 6.2 Effect of $LR_{G_1G_2}$

In Figure 3, we compare the effect of different values of $LR_{G_1G_2}$ on SQuAD-v2.0 dataset with BERT-large model. The ability to select a higher weightage of $G_1G_2$ is indeed effective, improving the performance of the model on both F1 and EM scores. Furthermore, the performance improvement is consistent over multiple orders of magnitudes of this parameter, from 0.1 to $1e^{-5}$, eliminating the need to fine-tune another hyper-parameter.

| **Method** | **F1** | **EM** |
|---|---|---|
| BERT | 80.6 ± 0.2 | 77.7 ± 0.2 |
| Reptile | 81.0 ± 0.2 | 78.0 ± 0.1 |
| FOMAML | 80.9 ± 0.1 | 77.9 ± 0.1 |
| DotProd | **81.9** ± 0.2 | **78.9** ± 0.2 |

Table 10: Comparison of our method to Reptile and FOMAML methods on SQuAD with BERT-large.

### 6.3 Effect of infrequent regularization

While algorithm 1 is first order, it introduces an overhead of 50% in computation. In order to minimize this overhead, instead of computing $G_1G_2$ for every two batches, an alternative is to use standard gradient updates for some batches, and only apply this regularization infrequently for a smaller number of batches. We study the effect of this on performance on SQuAD dataset with BERT-large model, by applying our method every 2, 3, 5, 8 and 10 batches, with overheads 50%, 33%, 20%, 12%, and 10% respectively. Even with only 10% over-

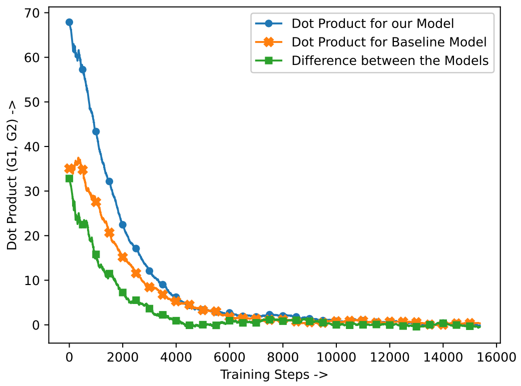

Figure 4: Dot Product between Gradients of Batches. Our algorithm significantly boosts the dot product between the batches, especially earlier in the training.

head, our regularization still results in significant performance gains of $1.0$ F1 and $0.9$ EM, with the improvement decreasing only slightly with reducing overhead.

### 6.4 Analysis of Gradient Dot Products

In Figure 4, we directly demonstrate that our method is indeed effective in boosting the dot-product between gradients while training. We compare two runs of BERT with the same seed on SQuAD-v2.0, and we plot the dot-product between gradients of batches. To minimize the effect of noise, we smoothen the plot by using a moving window of 100 batches, and remove outlier points more than $10\sigma$ away from the mean. During training, while dot-product naturally decreases to zero as the model converges (as also shown previously in Fort et al. (2019)), our approach significantly boosts the dot-product compared to the baseline, remaining consistently around $50 - 100\%$ larger the baseline throughout the training period.

### 6.5 Discussion of Effects on Training Dynamics

Training stability remains unaffected on all models/datasets we tried, and even improves slightly on Mini-Imagenet and Omniglot. Compared to our algorithm, the reptile algorithm appears unstable, perhaps due to larger $\alpha$. Our DotProduct method does not appear to make the model converge faster, with the rate of decrease of loss remaining almost identical to the baseline, but it does converge to a slightly lower loss. While the value of $LR_{G_1G_2}$, was kept fixed at $0.1$ in our experiments, model convergence remains unaffected upto around a value of $1$.

Higher values of this hyper-parameter may be helpful depending on the dataset. For example, we observed the scores on the QUASAR-Long dataset are ever higher with $LR_{G_1G_2}$ set at $0.5$, but we do not tune this parameter for different datasets in this paper. Also, while our algorithm is essentially the finite differences method to calculate the Hessian Vector product, we use the one-sided rather than centered version of finite differences to reduce compute overhead of our method.

## 7 Related Work

### 7.1 Transformer Models

Transformer models (Vaswani et al., 2017) are the backbone of most state-of-the-art NLP models. Models and pre-training techniques such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020) and Pegasus (Zhang et al., 2020) have made large improvements in the performance of NLP models. See subsection 4.2 for a detailed discussion of these models.

### 7.2 Meta-learning

Several works have explored Meta-learning, directly modify the learning process, such as by differentiating through the optimizer, such as MAML (Finn et al., 2017), Reptile (Nichol et al., 2018), Andrychowicz et al. (2016), Chen et al. (2017), giving performance improvements across a range of datasets and tasks. See subsection 2.1 for detailed descriptions of some meta-learning algorithms. While these approaches were initially proposed for few-shot multi-task learning, Kedia and Chinthakindi (2021) utilized the Reptile algorithm in single-task learning to improve generalization. Our approach is inspired from Reptile, but unlike Reptile, it gives us direct, explicit control over the importance of gradients' dot product.

### 7.3 Aligned Gradients

Previous works have explored alignments of gradients in the field of multi-task learning and continual learning, such as in Riemer et al. (2019), Lopez-Paz and Ranzato (2017), Chaudhry et al. (2018). Unlike these approaches, our method is First Order and does not require storing previously seen examples. Some recent works such as Fort et al. (2019) and Chatterjee (2020) also show that aligned gradients between examples is related to improved generalization and model performance.

PCGrad (Yu et al., 2020) proposes to minimize conflicting gradients by projecting only conflicting gradients to a normal place, while leaving aligned gradients unmodified, achieving significant improvements in multi-task supervised and RL tasks on image datasets. Fort et al. (2019) in particular, proposed gradient alignment as a meta-learning direction for future work, which this paper explores.

## 8 Conclusion

We propose to use finite-differences to calculate the gradient from the dot-product of gradients, and demonstrate its effectiveness as a regularization technique, leading to more aligned gradients between different batches. We leverage this approach to show performance improvements on several datasets such as SQuAD-v2.0, Quasar-T, and all the SuperGLUE datasets, and achieves state-of-the-art performance on Gigaword. Our method is effective over a range of models and model sizes, such as BERT, RoBERTa and Electra. Our method outperforms the Reptile and the FOMAML algorithm in single-task and few-shot multi-task settings, is first-order, is model-agnostic, and can be used with large models and large batches.

## References

Miltiadis Allamanis and Charles A. Sutton. 2013. Mining source code repositories at massive scale using language modeling. In *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*, pages 207–216. IEEE Computer Society.

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3981–3989.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. 2020. Second order optimization made practical. *CoRR*, abs/2002.09018.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST.

Satrajit Chatterjee. 2020. Coherent gradients: An approach to understanding generalization in gradient descent-based optimization. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, pages 556–572. Springer.

Yutian Chen, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matthew Botvinick, and Nando de Freitas. 2017. Learning to learn without gradient descent by gradient descent. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 748–756. PMLR.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pretraining text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23-2, pages 107–124.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern*

*Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.*, 29(6):141–142.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *CoRR*, abs/1707.03904.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All NLP tasks are generation tasks: A general pretraining framework. *CoRR*, abs/2103.10360.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Stanislav Fort, Pawel Krzysztof Nowak, and Srini Narayanan. 2019. Stiffness: A new perspective on generalization in neural networks. *CoRR*, abs/1901.09491.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

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*.

Akhil Kedia and Sai Chetan Chinthakindi. 2021. Keep learning: Self-supervised meta-learning for learning from inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 63–77, Online. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. 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 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. 2011. One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011*. cognitivesciencesociety.org.

Hector J. Levesque. 2011. The winograd schema challenge. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continuum learning. *CoRR*, abs/1706.08840.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Barak A. Pearlmutter. 1994. Fast exact multiplication by the hessian. *Neural Comput.*, 6(1):147–160.

Jonathan Pilault, Amine Elhattami, and Christopher J. Pal. 2020. Conditionally adaptive multi-task learning: Improving transfer learning in NLP using fewer parameters & less data. *CoRR*, abs/2009.09139.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. 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 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4603–4611. PMLR.

Gordon D Smith, Gordon D Smith, and Gordon Dennis Smith Smith. 1985. *Numerical solution of partial differential equations: finite difference methods*. Oxford university press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3630–3638.

Alex Wang, Ian F. Tenney, Yada Pruksachatkun, Phil Yeres, Jason Phang, Haokun Liu, Phu Mon Htut, , Katherin Yu, Jan Hula, Patrick Xia, Raghu Pappagari, Shuning Jin, R. Thomas McCoy, Roma Patel, Yinghui Huang, Edouard Grave, Najoung Kim, Thibault Févry, Berlin Chen, Nikita Nangia, Anhad Mohananey, Katharina Kann, Shikha Bordia, Nicolas Patry, David Benton, Ellie Pavlick, and Samuel R. Bowman. 2019. `jiant` 1.3: A software toolkit for research on general-purpose text understanding models. `http://jiant.info/`.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

# A Comparison to Previously Published Scores

## A.1 SQuAD

**BERT** The BERT paper reports scores of best-performing model. The mean score is a more robust measure, given the significant variation when fine-tuning BERT models, and hence we choose to report mean scores. Below are the results of our best performing model.

| **Method** | **F1** | **EM** |
|---|---|---|
| BERT (Devlin et al., 2019) | 81.9 | 78.7 |
| +DotProd | **83.16** | **80.28** |

Table 11: Comparison of our scores to published scores on SQuAD with BERT.

**RoBERTa** The RoBERTa paper reports a score of 86.5 EM and 89.4 F1 on SQuAD. Using the official parameters listed in the RoBERTa paper and their official checkpoint, while using the official SQuAD implementation of BERT, none of our 5 runs crossed 86.0 EM score. This is likely due to differences in handling unanswerable questions, long sequence length documents, etc. We report our reproduced scores in our main paper using the official SQuAD implementation of BERT with RoBERTa hyperparameters.

**ELECTRA** The Electra paper does not report SQuAD scores for Electra small. We report the reproduced scores using the official Electra github source code with default hyper-parameters. Note that Electra github reports a median score of 70.1 EM for Electra small, but none of our runs reached this performance, even on running 15 runs using fully official code and checkpoints.[1]

### A.2 SuperGLUE

Below we compare our reproduced BERT SuperGLUE scores to scores published in previous work.

| **Method** | **Score** |
|---|---|
| BERT (our paper) | 72.7 |
| BERT (Du et al., 2021) | 72.0 |
| BERT (Pilault et al., 2020) | 68.9 |
| BERT + DotProd | **73.9** |
| RoBERTa (our paper) | 79.6 |
| BERT (Du et al., 2021) | 81.5 |
| BERT (Pilault et al., 2020) | 76.5 |
| BERT + DotProd | **82.0** |

Table 12: Comparison of reproduced scores to published scores on SuperGLUE.

### A.3 Quasar-T

We only use the subset of the dataset in which the answer is an exact span, as mentioned in our main paper. As this is a non-standard subset, we report our reproduced scores.

### A.4 Gigaword

We report the official scores from Pegasus github for “Mixed & Stochastic” model as our baseline. Note that these github scores are higher than those reported in the Pegasus paper.

### A.5 Omniglot and MiniImageNet

We report the official scores from the Reptile and MAML papers.

## B Links to Source code

For SuperGLUE, we use the Official Implementation for BERT and RoBERTa available at https://github.com/nyu-mll/jiant, along with the default pre-trained models.

For SQuAD, QUASAR and NewsQA, we used the official implementation and pre-trained models at https://github.com/google-research/bert for BERT and the official pre-trained models from https://github.com/pytorch/fairseq/tree/master/examples/roberta for RoBERTa.

For Pegasus, we used the official implementation and “Mixed & Stochastic” pre-trained model weights at https://github.com/google-research/pegasus.

For Omniglot and Mini-Imagenet, we used the official code from Reptile here https://github.com/openai/supervised-reptile

The DotProd Optimizer is trivial to implement in all of the above models following the pseudocode from our the main paper, by modifying the *Optimizer* class used for each of the models.

## C Links to Download Data

SuperGLUE can be downloaded from https://super.gluebenchmark.com/.

SQuAD v2.0 can be downloaded from https://rajpurkar.github.io/SQuAD-explorer/.

GigaWord can be downloaded using https://www.tensorflow.org/datasets/catalog/gigaword/.

QUASAR-T can be downloaded from https://github.com/bdhingra/quasar.

Omniglot can be downloaded from https://github.com/brendenlake/omniglot/tree/master/python.

Mini-imagenet can be downloaded following instructions from https://github.com/yaoyao-liu/mini-imagenet-tools.

[1] The command used was - python3 run_finetuning.py –data-dir DATADIR –model-name electra_small –hparams ‘{"model_size": "small", "task_names": ["squad"]}’

## D Evaluation Metric code

We used the original evaluation metrics code for all our models, available from source code and datasets linked above.

## E Dataset Details and Evaluation Metrics

### E.1 SuperGLUE

**BoolQ** Boolean Questions, a Question Answering (QA) dataset with short passages and yes/no questions, with data from Wikipedia and Google search engine queries.

**CB** Commitment Bank, consisting of passages with labels for *commitment* of speakers of clauses to said clause, framed as three-class NLI, with data from WSJ, British National Corpus and SwitchBoard. Evaluated with unweighted average F1 and accuracy.

**COPA** Choice of Possible Alternative, a dataset to classify the cause/effect of a given premise from two alternatives, with fully handcrafted data.

**MultiRC** Multi-Sentence Reading Comprehension, a QA dataset, with a list of multiple-choice possible answers for each question to a paragraph. Evaluated with F1 over all answer-options($F1_a$), and exact match of each question's set of answers($EM$).

**ReCoRD** Reading Comprehension with Commonsense Reasoning, a QA dataset consisting of articles and Cloze-style questions with a masked entity, scored on predicting the masked entity from the entities in the article, with data from CNN and Daily Mail. Scored with token-level F1 and EM.

**RTE** Recognizing Textual Entailment, as binary classification of *entailment* or *not entailment*, with data from Wikipedia and news.

**WiC** Word-in-Context, a word sense disambiguation (WSD) dataset, tasked with binary classification of sentence pairs based on the sense of a common polysemous word. Data is from WordNet and Wiktionary.

**WSC** Winograd Schema Challenge, a coreference resolution task on resolving pronouns to a list of noun phrases. As the models we tested only predicted the majority class, we omit this dataset.

### E.2 SQuAD v2.0

The Stanford Question Answering Dataset v2.0 is a popular span-style QA dataset, consisting of passages from Wikipedia, labelled by annotators for questions on the passages and corresponding answer spans, along with unanswerable questions as well. This dataset is evaluated with F1 and EM scores of predicted answer spans.

### E.3 GigaWord

Gigaword is a summarization dataset, with single-line input documents from news sources, and task is to generate headlines. The dataset is pre-tokenized and number are replaced with #. Evaluation is using ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004) metrics.

### E.4 Quasar-T

QUASAR-T is a large-scale dataset aimed at evaluating systems designed to comprehend a natural language query and extract its answer from a large corpus of text. It consists of open-domain trivia questions and their answers obtained from various internet sources. We only use those questions whose answers can be extracted as a span for our training and evaluation.

## F Label Distributions for datasets

**SuperGLUE datasets** - The baseline scores by always predicting the most frequent class are 62.3 accuracy for BoolQ, 21.7/48.4 Avg. F1 / Accuracy for CB, 50.0 accuracy for COPA, 61.1/0.3 F1a / EM for MultiRC, 33.4/32.5 F1 / Accuracy for ReCoRD, 50.3 accuracy for RTE, and 50.0 accuracy for WiC.

**SQuAD v2.0** train set has a total of 130,319 questions of which 43,498 are unanswerable, whereas the dev set has a total of 11,873 questions of which 5,945 are unanswerable. The answer-span location varies across the input.

**NewsQA** train set has a total of 97,313 questions of which 20,753 are unanswerable, whereas the dev set has a total of 5,456 questions of which 1,115 are unanswerable. The answer-span location varies across the input.

**Quasar-T Long** train set has 24,499 questions whereas the dev set contains 1,920 questions. The answer-span location varies across the input.

**Quasar-T Short** train set has 20,533 questions whereas the dev set contains 1,653 questions. The answer-span location varies across the input.

**Gigaword** train set has 3,803,957 articles, dev set has 189,651 articles, and test set has 1,951 articles for summarization.

**Omniglot** has 1200 classes in Train, 423 in Test with 20 images per class.

**Mini-Imagenet** has 64 classes in Train, 20 in Test, with 600 images per class.