Adversarial Training in Distillation Of BERT

Vijay Kalmath
Department of Data Science
Columbia University
New York, NY 10027
vsk2123@columbia.edu

Amrutha Varshini Sundar
Department of Data Science
Columbia University
New York, NY 10027
as6431@columbia.edu

Sean Chen
Department of Computer Science
Columbia University
New York, NY 10027
sean.chen@columbia.edu

Abstract

In the recent years, transformer-based language learning models like BERT have been one of the most popular architectures used in the research related to natural language processing. Productionizing these models under constrained resources such as edge computing require smaller versions of BERT like DistilBERT. However, it is a concern that these models have inadequate robustness against adversarial examples and attacks. This paper evaluates the performance of various models built on the ideas of adversarial training and GAN BERT finetuned on SST-2 dataset. Further the experiments in this paper seek to find evidence on whether knowledege distillation preserves robustness in the student models.

1 Introduction

Over the past years, BERT-like models have been dominating the research of natural language process and deep learning. Distilled version of BERT and GAN-BERT have also been developed to expand the scope of research approaches. However, it should cause attention and concern to ensure the adversarial robustness and reliability of those models. Therefore, in this paper, we take advantage of an existing Python framework, TextAttack, to investigate the effects and performance of BERT-like models under the adversarial attacks. As of our research targets, we build and train and fine-tune a DistilBERT model, then train a GAN-BERT model and a distilled version of GAN-BERT. We test and base our attack on two main categories of adversarial attack algorithms: visual adversarial attack algorithms and context-based adversarial attack algorithms.[Jacob Devlin, 2018]

2 Related Works

2.1 Pre-training Language Representation Approaches

Word embedding pre-training has become a vital tool for modern natural language processing research. There are unsupervised feature-based and fine-tuning approaches in the downstream tasks. [Jacob Devlin, 2018] The feature-based method extracts bidirectional contextual features. The fine-tuning method utilizes the the pre-trained models to fine-tune all parameters in which very few have to be learned from scratch.

Source code can be accessed at https://github.com/VijayKalmath/AdversarialTraining_in_Distilation_Of_BERT

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

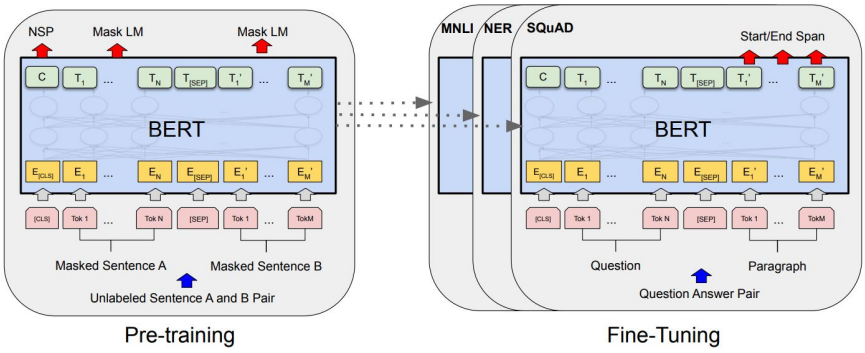

Figure 1: Pre-training and Fine-Tuning of BERT

2.2 BERT and DistilBERT

The Bidirectional Encoder Representations from Transformers comprise two steps. First, the pre-training trains the model on the unlabeled data. Second, the fine-tuning steps takes labeled data to tune all of the pre-trained parameter for the downstream tasks, as illustrated in Figure 1. The input embeddings of BERT consists of the token embeddings, segment embeddings, and position embeddings. DistilBert is a lightweighted BERT model with 40% of the original weight and 60 % faster. With the token embeddings and the pooler removed, the DistilBERT only have half of the layers as BERT. [Victor SANH, 2020]

2.3 Adversarial Training and Model Robustness

Adversarial Training is a training method to intrinsically enhance the robustness of deep learning models and defend against adversarial examples. Those examples are generated into the training data for the model to adapt to the maliciousness and become more robust.Afterwards we evaluate the robustness of a model based on adversarial attacks with test data and the accuracy results. Model robustness is a term to measure the ability of a model to defend against adversarial attacks.[Xin Guo, 2021] has provided a function that defines robustness in terms of student model, original data-set and accuracy of student model on the dataset, the generated adversarial set, neighbourhood of sample and loss function for the attacks to maximize the attacks.

2.4 TextAttack

TextAttack is Python framework to launch adversarial attacks, enable data augmentation and implement adversarial training for natural language process models. It can launch model-specific and evaluate the results, and improve robustness in the downstream and model generalization. There are four parts in the attacks. First, a goal function measures the success of the attack based on the output. Second, constraints determines whether a perturbation of the attack is valid. Third, transformation produces perturbation based on an input during the attack. Fourth, a search method picks the best perturbations from the transformation.[Morris et al., 2020]

3 Experiments

Language models like BERT and other variants have taken over the sequence modelling domain and their ubiquitous usage makes it all the more important for their robustness. The pith of our experimentation in building several models is an attempt to validate and improve the robustness of these models through different methods. Along with robustness, the other feature under consideration is model usability in terms of model size and computational overhead in edge applications. As bulky models like BERT are not fit for constrained environment, we explore delivering the capabilities of robustness from smaller models such as DistilBERT. On all the following experimentation the dataset used is SST-2 from the GLUE benchmark.

3.1 Problem Formulation

In this section we formulate our problem statement with respect to building BERT like models that are robust and less complex for usage. We explore two basic strategies for building robust models that are immune to adversarial attacks - Adversarial training and GAN-Bert training. Adversarial training mainly involves the technique of data augmentation during the finetuning step of the model while GAN Bert training involves self creation of perturbed examples that are included during the model training phase. Once these models have been trained, we aim to discover whether distillation methods such as student-teacher distillation preserves the robustness to adversarial attack from the teacher to the student.

3.2 Experiment Settings

3.2.1 Finetuning pre-trained DistilBERT

The main idea of this section is to train and build and fine tuned DistilBERT model from the pretrained DistilBERT available from the HuggingFace library. As claimed by works in Xin Guo [2021], BERT and the process of distillation doesn't preserve the robustness of the parent models. This model is created in the intention of validating the above claims. For the purposes of fine tuning, the pretrained DistilBERT model chosen is DistilBertForSequenceClassification from the HuggingFace library which is adapted for binary sentiment classification as required for the SST2 dataset. The number of labeled examples trained on from the training set is 67,349 which are either positive or negative movie reviews.

The training was performed with a batch size of 64 and for 10 epochs followed by the Adam optimizer with a learning rate of $2e^{-5}$ and a standard binary cross entropy loss. The training approximately took 18 minutes on a single V100 GPU instance. After completion of the fine tuning module, the overall reported accuracy was observed to be 89.9%.

3.2.2 Training GAN-Bert

As one of the steps towards building robust models, the identified approach is building a GAN Bert influenced by ideas from Croce et al. [2020]. The architecture involves embedding a pretrained BERT model within two multi layer perceptron layers that act as the generator and discriminator respectively. The usage of the GAN modules inherently trains the model for adversarial examples by adding a small noise component to the input training examples.

As suggested by the original paper, this architecture was built to check the robustness of the model in settings where there are very less labeled examples available. In order to recreate the a similar situation, we synthetically created three different training sets with 0.2%, 0.5% and 0.7% labeled set whereas the others were labeled with the category as unknown - "UNK".

The generator and the discriminator are functionally similar to General Adversarial Networks idea as seen in Ian J. Goodfellow [2014]. In this case, the generator model adds some small perturbations in the form of noise to the original input text from the training set while the discriminator is set to classify the incoming example during training into $k + 2$ classes where the last two classes stand for 'UNK' and 'FAKE' example.

Architecturally, the generator and the discriminator are one layer MLP with size equal to the number of hidden layers of the pretrained BERT model which in our case is 768 neurons. Additionally the discriminator has a final output layer consisting of 4 neurons which each represents the following classes - [positive, negative, 'UNK', 'FAKE']. The generator loss consist of two components - one for the output from the descriminator on fake examples and the other acting as the regularizer.

$$G_{loss} = -\frac{\sum(\log(1 - P(fake)_D + \epsilon))}{n} + reg$$

The discriminator loss is dependent on whether the fed example is from the real training dataset or whether it was perturbed by the generator. Thus the discriminator loss while similar to the generator loss has the following structure where real is a binary flag which stands for whether the example was real or not.

$$D_{loss} = real \times loss_{supervised} + (1 - real) \times loss_{unsupervised}$$

The training was performed with a batch size of 64 and for 10 epochs followed by the Adam optimizer with a learning rate of $2e^{-5}$ and the generator discriminator loss described above. The training approximately took 59 minutes on a single V100 GPU instance.

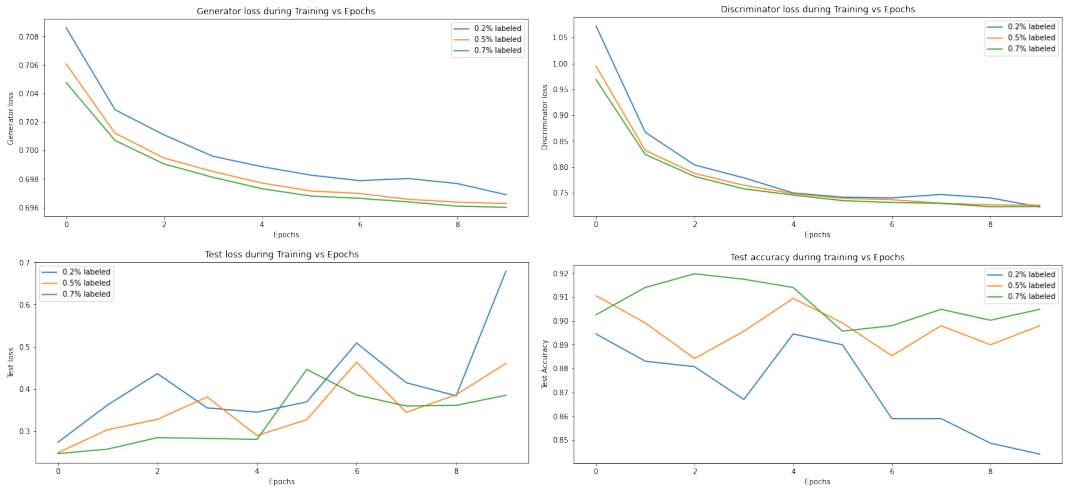

Figure 2: GanBert Training Metrics

3.2.3 Adversarial training using data augmentation

Adversarial Data Augmentation Method	training_accuracy	validation_accuracy	Training_time(in mins)
SST-2 + EmbeddingAugmenter	0.99	0.89	28
SST-2 + SynonymInsertionAugmenter	0.99	0.89	28
Only SynonymInsertionAugmenter	0.99	0.89	10
Only EmbeddingAugmenter	0.99	0.88	19
Only WordNetAugmenter	0.99	0.88	10
Only AllData	0.99	0.87	31
SST-2 + WordNetAugmenter	0.99	0.87	28

Table 1: Training and Validation Accuracy on SST-2 + Adversarial Data Augmentation for 6 Models.

Standard Data Augmentation from the image processing domain has always known to improve the Model's capabilities to generalize better in the image processing domain.Previous work has show Data augmentation through synthesis of fake inputs also has a similar intended affect on NLP and Transformer based models.

Adversarial data augmentation for NLP and BERT has emerged as a strong baseline for data augmentation methods in recent work [Ganin et al. [2015], Volpi et al. [2018],Sinha et al. [2017]] which are focused on increasing robustness of the deep learning nlp models .

Amongst the the various techniques generate adversarial data from the training dataset , techniques can be bifurcated as a visual data augmentation technique where characters are swapped or words are deleted in random or as a semantic data augmentation technique where words are replaced with another candidate words based on similarity to maintain the meaning and the context of the sentence. While visual data augmentations are important and do impact robustness of models these techniques often result in unnatural looking adversarial examples which lack grammatical correctness, In our experiments we focus on contextual aware data augmentation .

In our experiments, we use 3 main data augmentation techniques all built around the central idea of swapping words in the sentence with other words while maintaining the context of the sentence with the SST-2 datasets.

The EmbeddingAugmenter technique replaces random words in a sentence with its word neighbors in the Embedding space whereas the SynonymInsertion Augmenter and WordNetAugmenter find synonyms for the words in the input sentence from different thesauruses to create synthetic training examples.

From the 60K training examples in the SST-2 train dataset , we randomly choose 20K examples and generate 2 data augmented examples for each of the sentences with an average perturbation of 20%. We aggregate 40K adversarial examples from each of the three different adversarial data augmentation techniques for the adversarial training.

With the different data augmentation datasets created, to explore the different ways to use them in training the distilbert we used to different techniques.

In the first technique , we take a pre-trained BERT from hugging-face , append one of the new synthetic datasets to the original SST-2 training dataset and perform fine-tuning based in similar fashion to the steps described in section 3.2.1 .

In the latter technique , we load the fine-tuned distilbert that was created in the section 3.2.1 and train it only on the new synthetic datasets for 10 Epochs .

With 3 different data augmentation techniques and 2 different types of training the distilbert with them,we generated 6 different models and use them in the further steps to understand which augmentation performed better and which technique out-performs the other.

We also train a model with all the 3 data augmentation examples combined to check if using a mixture of data adversarial augmentation works better than using individual augmentation techniques.

From manual checks of the augmented data and the validation accuracy of the SST2+ EmbeddingAugmenter model, we believe that the Embedding Augmenter is a better choice amongst the augmenters we used in this analysis.

3.2.4 Distillation of GAN-Bert

In order to answer the question of whether distillation preserves the robustness from the teacher model, we create a distilled version of the GAN Bert. Based on the idea that comes from Victor Sanh [2019] where it was shown that a student performed better than simply finetuning the distilled language model, we are performing a task specific knowledge distillation from the teacher model.

The loss function used at the student model is a weighted sum of the KL divergence loss between the logits from the Finetuned teacher and the Half finetuned DistilBERT student model coupled with inherent model loss from student distilbert model. Log softmax activation along with temperature is applied for the logits coming from the Finetuned teacher and DistilBERT model.

$$Loss_{logit} = KL(log(softmax(\frac{logit_S}{T})), softmax(\frac{logit_T}{T}))$$

$$Loss_{distil} = \alpha \times loss_S + (1 - \alpha) \times Loss_{logit}$$

The distillation was performed with a batch size of 128 and for 7 epochs followed by the Adam optimizer with a learning rate of $6e^{-5}$ and with the parameters $\alpha = 0.5$ and temperature $T = 4$. The training approximately took 1 hour 9 minutes on a single V100 GPU instance. After completion of the distillation of the GAN Bert module, the validation loss and accuracy were found to be as shown in the plots.

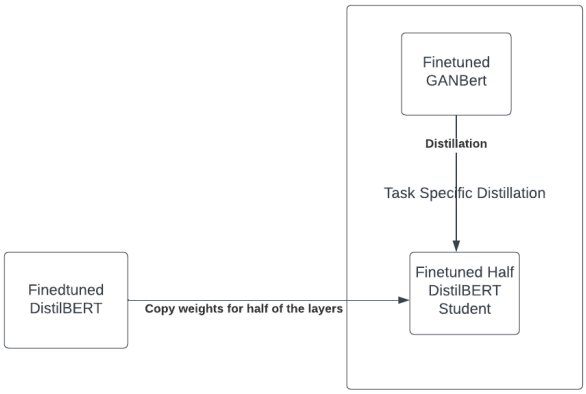

Figure 3: Task Specific Distillation of GANBert

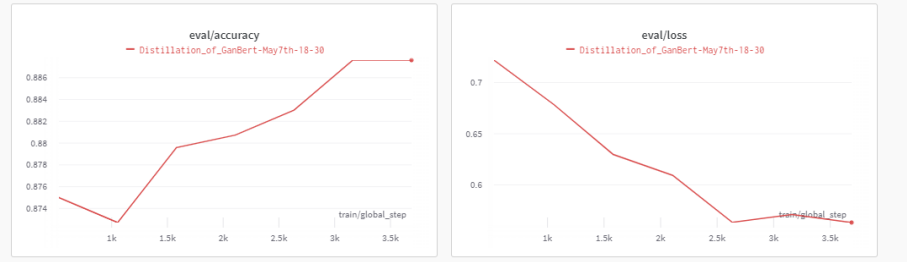

Figure 4: Model evaluation loss and accuracy on test set vs Training Step

3.2.5 Adversarial attack using TextAttack

Applying imperceptible non-random perturbations to a test entity to change the model's prediction or in other words to trick a model has been well studied since the seminal work by Szegedy et al. [2013]. In the NLP domain these perturbations come in the form on changing words with there synonyms or using double negative contexts to trick the model to changing its prediction. In Jin et al. [2019] , they use this concept of adversarial attacks to ask and understand the actual robustness of pre-trained BERT and its use in domains like sentence classification or sentence entailment. Our experiments are influenced by their work as we try to understand the affects of distillation to the robustness

Adversarial attacks in NLP can be broadly divided into two parts based on the extinct of access to the model that the "attacker" has. In Black Box adversarial settings the attacker does not any knowledge of the model's parameter space , gradient space and its architecture, the attack can only interact with the inputs of the model and get the results and confidence scores from the model. Conversely in white box adversarial settings , the model's details are made available to the attacker along with the weights and gradients in order for a more specific attack on the model. In our experimentation , we only use the black box adversarial setting as it is more commonplace in the world than white box.

We use the textAttack framework created by Morris et al. [2020] to perform adversarial attacks on our BERT based models in order to analyze their robustness to understand how a model's training and architecture affects the robustness.

The textAttack framework is an aggregation of adversarial attack algorithms in the NLP domain. After understanding the various adversarial attacks , we decided to use 4 different adversarial attack algorithms to test the robustness of our models.

The four different adversarial attack Algorithms we are using are

1. Visual Adversarial Attack Algorithms
   (a) Gao et al. [2018] - Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers DeepWordBug generates small text perturbations in a similar black box adversarial setting which forces a text classifier to misclassify a text input . It uses four scoring functions to score the importance of the tokens in the input

sequence. It then based on Levenshtein distance (edit distance) constraint creates small transformations in the input sequence.

(b) Pruthi et al. [2019] - Combating Adversarial Misspellings with Robust Word Recognition. Pruthi is also a black box attack algorithm which performs single individual character attacks followed by 2 character attacks and so on for higher order attacks. The algorithm performs a greedy search and does not use Greedy-WIR search methodology but rather a vanilla Greedy Search. The perturbations performed by this attack type are swapping , QWERTY key substitutions to mimic misspellings etc..

2. Context Based Adversarial Attack Algorithms

(a) Jin et al. [2019] TextFooler - black box adversarial setting attack algorithm uses the Greedy-WIR (Word Importance Ranking) methodology to choose words that need to be transformed for the adversarial attack.The importance score given to a word is calculated as the prediction change before and after deleting a word in the input sentence . With the importance score , the choosen word is transformed using a synonym extractor and a POS checker to weed out transformation candidates who are not in the right POS as the original word.

(b) Garg and Ramakrishnan [2020] - BERT-based Adversarial Examples s BAE is a black box adversarial setting attack algorithm that uses contextual perturbations from a BERT masked language model. It replaces and inserts tokens by masking a portion of the text and using BERT-MLM to generate alternatives for the masked tokens.In order to replace the most important tokens and to choosen the most weighted token from the results of the BERT-MLM , BAE uses a Greedy-WIR algorithm .

4 Results

Model compression or distillation has become common-place in all NLP deep learning model deployments since Victor Sanh [2019] proved there efficacy and the minimal drop in accuracy. Our work traced the different ways adversarial training can be applied to the distillation pipeline in order to make the final models more robust and impervious to adversarial attacks.

As seen from 4, the gan-bert model has its evaluation accuracy in the similar range of as the distilbert even when the ganbert model does not have access to all the training dataset. This implies that the Discriminator and Generator in the Gan-Bert are able to work together well to model the input distribution.

The table 4 shows that there is a trade-off between standard Evaluation accuracy and imperviousness to adversarial attacks in the gan-bert models.We see that the ganbert model with 0.5 labelled data has the best performance under attack amongst the 4 Models which the ganbert model trained on 0.7 labelled data has the best evaluation accuracy. It is interesting to note that increase in labelled examples in the dataset does not increase the robustness of the ganbert model but infact has a point of diminishing returns, we believe this is because the generator and discriminator do not get to play much role in the model when there is an abundance of data.

Based on the results of distilled models and their behavior with respect to the teacher models as seen in 4

			Accuracy under Attack				Attack Success Rate			
Bert	Label ratio	Accuracy	BAE	DeepW	Pruthi	TextFooler	BAE	DeepW	Pruthi	TextFooler
distilbert	1	90.25	32.24	12.42	47.39	3.84	64.29	86.27	47.52	95.80
ganbert	0.2	84.4	29.82	15.25	44.61	2.64	64.67	81.93	47.15	96.88
	0.5	89.79	36.35	16.51	47.59	5.85	59.51	81.61	47.0	93.49
	0.7	90.48	38.19	12.27	50.69	2.98	57.79	86.44	43.98	96.7

Table 2: Adversarial Attacks on Standard Fine-Tuned Models

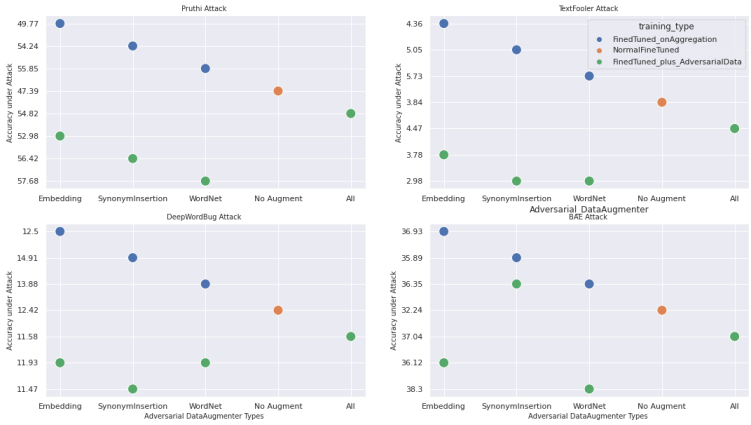

Figure 5: Task Specific Distillation of GANBert

From the above graphs ,We can see that all types of Adversarial data augmentation to the original dataset increases the robustness of the model but only when the synthetic data is appended to the original SST-2 dataset prior to training.Fine-tuning the distilled model with just the adversarial examples seems to be very detrimental to both overall accuracy and accuracy under adversarial attacks.The Embedding Augmenter performs the best under Adversarial Attacks among the rest indicating that the model becomes more robust with slight changes in the embedding domain.

Adversarial Attack Algorithm	Avg Accuracy underAttack	Avg Attack SuccessRate	Avg PerturbedWord	Avg Queries
TextFooler	3.83	95.69	17.30	91.70
DeepWordBug	12.87	85.48	19.81	33.94
BAE	35.60	59.86	13.16	60.24
Pruthi	51.81	41.57	8.41	323.74

Table 3: Performance of the different Adversarial Attacks

The efficacy of different adversarial attacks can be seen from 4 which indicates that the semantic based TextFooler model is extermely hard to defend whereas on an average the models built so far are able to defend against the visual based Pruthi attacks achieving accuracy of 50%.

It is also interesting to note the number of Queries performed for each adversarial attack algorithm for an attack , as TextFooler, DeepWordBug and BAE all use Greedy-WIR search method , the average number of queries to get the adversarial input sequence is far lower than Pruthi which is based on a standard Greedy Search method.

			Accuracy Under Attack				Attack Success Rate				Avg PerturbedWord				Avg Queries			
Bert Model	Distilled	text_attack_recipe Original Accuracy	BAE	DeepWordBug	Pruthi	TextFooler	BAE	DeepWordBug	Pruthi	TextFooler	BAE	DeepWordBug	Pruthi	TextFooler	BAE	DeepWordBug	Pruthi	TextFooler
distilbert	No	90.25	32.24	12.42	47.39	3.84	64.29	86.27	47.52	95.80	13.00	18.55	8.34	17.10	62.00	34.16	323.80	92.09
	Yes	88.76	33.60	9.75	49.66	1.26	62.14	89.02	44.06	98.58	13.27	19.55	8.50	16.22	59.61	32.00	322.64	80.55
gan-bert	No	89.79	36.35	16.51	47.59	5.85	59.51	81.61	47.00	93.49	12.88	19.09	8.08	17.07	61.21	34.57	325.89	95.27

Table 4: Performance of Distilled Model with respect to its teachers

The table 4 shows the affect of distillation on the model's robustness , we can see that the student model performs very bad with respect to it teacher (gan-bert) and the parent distilbert from where the weights are copied. We can see that even with half of the layers in student model , we can see that the original accuracy is very close to the teacher's accuracy which maps with the work in DistilBert , but the drop in robustness is extremely high. This follows our initial hypothesis that distillation does not carry robustness of the teacher onto the student and adversarial finetuning is extremely important.

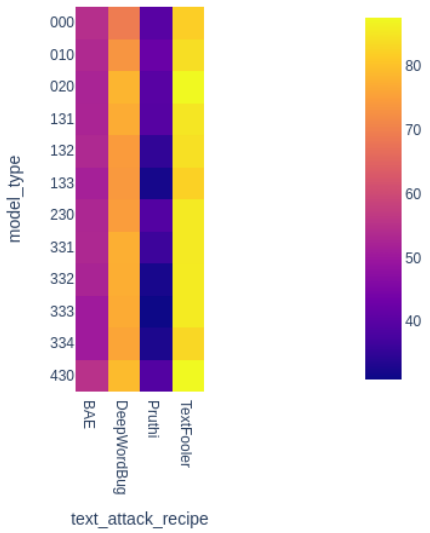

Figure 6: Accuracy drop for different model across attack recipes

We are also interested in the overall robustness and resistance to adversarial attack performance review for all the models built using the techniques of adversarial training and model distillation. A comprehensive heatmap representation gives a visualization of drop in accuracy under attack by different model recipes 4. The model convention per digit from left to write stand for the following,
[GAN, DistilBERT Finetuned with data augmentation, DistilBERT Finetuned, Finetuned DistilBERT train on Augmented data, GANBert Distilled] in the first digit,
[0.2 drop in labeled examples, 0.5 drop in labeled examples, 0.7 drop in labeled examples, No drop in labeled examples] in the second digit and
[No Augmentation, Embedding Augmentation, Synonym Augmentation, WordNet Augmentation, All types of augmentation] in the third.

We can observe that an overall trend in drop in accuracy under attack from the original holds for all the model in the following increasing order of attack recipes - Pruthi, BAE, DeepWordBug and TextFooler with TextFooler being the strongest of the attackers.

- Comparing the models 230 - Finedtuned DistilBERT on SST-2 to and 334 - Finedtuned distil model with all augmentations, the later is able to defend much better as it has a relatively lower accuracy drop and hence it is certainly beneficially to create robust models by finetuning the model with data augmentation.
- Now the same model 334 in comparison to simple 000 - GanBert with 0.2 drop of labeled examples seems to have a higher accuracy drop for DeepWordBug, Pruthi and TextFooler. This indicates that GanBert model is much more powerful than adversarial training of FineTuned Distil models.
- Infering results for the question of whether distilation preserves robustness, we compare the models 430 - GAN Bert distilled model and 010 - the original GANBert model with 0.5 drop in labeled examples, we clearly see the accuracy drop with all recipes and is most significant for TextFooler and DeepWordBug. This provides an evidence that our experimentation showed that distillation from teacher model doesn't preserve the robustness in the student model.

References

Danilo Croce, Giuseppe Castellucci, and Roberto Basili. GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2114–2119, Online, July 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.acl-main.191.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. arXiv, 2015. doi: 10.48550/ARXIV.1505.07818. URL https://arxiv.org/abs/1505.07818.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers, 2018. URL https://arxiv.org/abs/1801.04354.

Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification, 2020. URL https://arxiv.org/abs/2004.01970.

Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial networks. arXiv, 2014. doi: https://doi.org/10.48550/arXiv.1406.2661. URL https://arxiv.org/abs/1406.2661.

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

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. arXiv, 2019. doi: 10.48550/ARXIV.1907.11932. URL https://arxiv.org/abs/1907.11932.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. arXiv, 2020. doi: 10.48550/ARXIV.2005.05909. URL https://arxiv.org/abs/2005.05909.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. Combating adversarial misspellings with robust word recognition, 2019. URL https://arxiv.org/abs/1905.11268.

Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying some distributional robustness with principled adversarial training. arXiv, 2017. doi: 10.48550/ARXIV.1710.10571. URL https://arxiv.org/abs/1710.10571.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv, 2013. doi: 10.48550/ARXIV.1312.6199. URL https://arxiv.org/abs/1312.6199.

Julien Chaumond Thomas Wolf Victor Sanh, Lysandre Debut. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv, 2019. doi: https://doi.org/10.48550/arXiv.1910.01108. URL https://arxiv.org/abs/1910.01108.

Julien CHAUMOND Thomas WOLF Victor SANH, Lysandre DEBUT. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. 2020.

Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/1d94108e907bb8311d8802b48fd54b4a-Paper.pdf.

Haoyi Zhou Xucheng Ye Jianxin Li Xin Guo, Jianlei Yang. Rosearch: Search for robust student architectures when distilling pre-trained language models. pages 5152–5161. sc.CL, 2021.