Universal Language Model Fine-tuning for Text Classification

Jeremy Howard*
fast.ai
University of San Francisco
j@fast.ai

Sebastian Ruder*
Insight Centre, NUI Galway
Aylien Ltd., Dublin
sebastian@ruder.io

Abstract

Inductive transfer learning has greatly impacted computer vision, but existing approaches in NLP still require task-specific modifications and training from scratch. We propose Universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP, and introduce techniques that are key for fine-tuning a language model. Our method significantly outperforms the state-of-the-art on six text classification tasks, reducing the error by 18-24% on the majority of datasets. Furthermore, with only 100 labeled examples, it matches the performance of training from scratch on 100× more data. We open-source our pretrained models and code[1].

1 Introduction

Inductive transfer learning has had a large impact on computer vision (CV). Applied CV models (including object detection, classification, and segmentation) are rarely trained from scratch, but instead are fine-tuned from models that have been pretrained on ImageNet, MS-COCO, and other datasets (Sharif Razavian et al., 2014; Long et al., 2015a; He et al., 2016; Huang et al., 2017).

Text classification is a category of Natural Language Processing (NLP) tasks with real-world applications such as spam, fraud, and bot detection (Jindal and Liu, 2007; Ngai et al., 2011; Chu et al., 2012), emergency response (Caragea et al., 2011), and commercial document classification, such as for legal discovery (Roitblat et al., 2010).

While Deep Learning models have achieved state-of-the-art on many NLP tasks, these models are trained from scratch, requiring large datasets, and days to converge. Research in NLP focused mostly on *transductive* transfer (Blitzer et al., 2007). For *inductive* transfer, fine-tuning pretrained word embeddings (Mikolov et al., 2013), a simple transfer technique that only targets a model's first layer, has had a large impact in practice and is used in most state-of-the-art models. Recent approaches that concatenate embeddings derived from other tasks with the input at different layers (Peters et al., 2017; McCann et al., 2017; Peters et al., 2018) still train the main task model from scratch and treat pretrained embeddings as fixed parameters, limiting their usefulness.

In light of the benefits of pretraining (Erhan et al., 2010), we should be able to do better than *randomly initializing* the remaining parameters of our models. However, inductive transfer via fine-tuning has been unsuccessful for NLP (Mou et al., 2016). Dai and Le (2015) first proposed fine-tuning a language model (LM) but require millions of in-domain documents to achieve good performance, which severely limits its applicability.

We show that not the idea of LM fine-tuning but our lack of knowledge of how to train them effectively has been hindering wider adoption. LMs overfit to small datasets and suffered catastrophic forgetting when fine-tuned with a classifier. Compared to CV, NLP models are typically more shallow and thus require different fine-tuning methods.

We propose a new method, Universal Language Model Fine-tuning (ULMFiT) that addresses these issues and enables robust inductive transfer learning for any NLP task, akin to fine-tuning ImageNet models: The same 3-layer LSTM architecture—with the same hyperparameters and no additions other than tuned dropout hyperparameters—outperforms highly engineered models and trans-

[1] http://nlp.fast.ai/ulmfit.

*Equal contribution. Jeremy focused on the algorithm development and implementation, Sebastian focused on the experiments and writing.

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 328–339
Melbourne, Australia, July 15 - 20, 2018. ©2018 Association for Computational Linguistics

fer learning approaches on six widely studied text classification tasks. On IMDb, with 100 labeled examples, ULMFiT matches the performance of training from scratch with 10× and—given 50k unlabeled examples—with 100× more data.

Contributions Our contributions are the following: 1) We propose Universal Language Model Fine-tuning (ULMFiT), a method that can be used to achieve CV-like transfer learning for any task for NLP. 2) We propose *discriminative fine-tuning*, *slanted triangular learning rates*, and *gradual unfreezing*, novel techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning. 3) We significantly outperform the state-of-the-art on six representative text classification datasets, with an error reduction of 18-24% on the majority of datasets. 4) We show that our method enables extremely sample-efficient transfer learning and perform an extensive ablation analysis. 5) We make the pretrained models and our code available to enable wider adoption.

2 Related work

Transfer learning in CV Features in deep neural networks in CV have been observed to transition from task-*specific* to *general* from the first to the last layer (Yosinski et al., 2014). For this reason, most work in CV focuses on transferring the last layers of the model (Long et al., 2015b). Sharif Razavian et al. (2014) achieve state-of-the-art results using features of an ImageNet model as input to a simple classifier. In recent years, this approach has been superseded by fine-tuning either the last (Donahue et al., 2014) or several of the last layers of a pretrained model and leaving the remaining layers frozen (Long et al., 2015a).

Hypercolumns In NLP, only recently have methods been proposed that go beyond transferring word embeddings. The prevailing approach is to pretrain embeddings that capture additional context via other tasks. Embeddings at different levels are then used as features, concatenated either with the word embeddings or with the inputs at intermediate layers. This method is known as hypercolumns (Hariharan et al., 2015) in CV[2] and is used by Peters et al. (2017), Peters et al. (2018), Wieting and Gimpel (2017), Conneau et al. (2017), and McCann et al. (2017) who use language modeling, paraphrasing, entailment, and Machine Translation (MT) respectively for pretraining. Specifically, Peters et al. (2018) require engineered custom architectures, while we show state-of-the-art performance with the same basic architecture across a range of tasks. In CV, hypercolumns have been nearly entirely superseded by end-to-end fine-tuning (Long et al., 2015a).

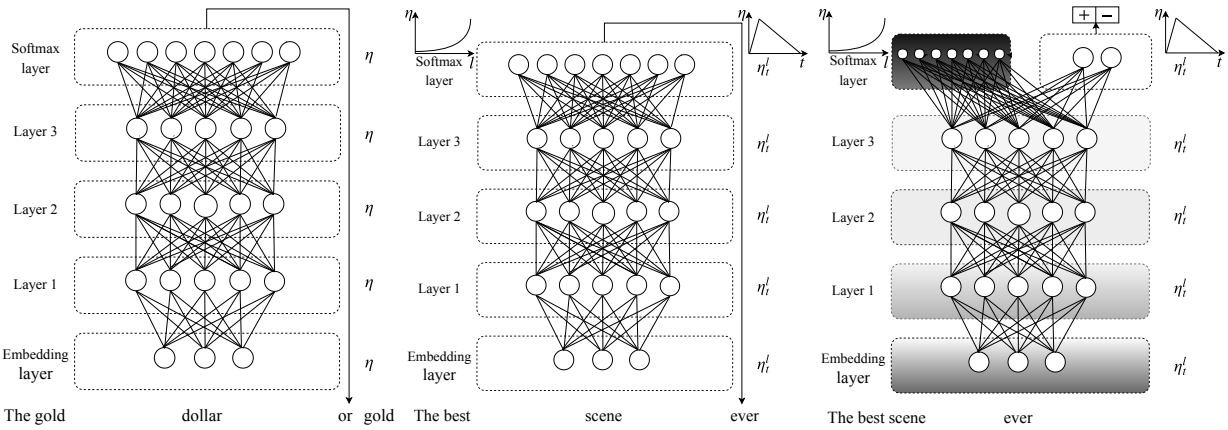
Multi-task learning A related direction is multi-task learning (MTL) (Caruana, 1993). This is the approach taken by Rei (2017) and Liu et al. (2018) who add a language modeling objective to the model that is trained jointly with the main task model. MTL requires the tasks to be trained from scratch every time, which makes it inefficient and often requires careful weighting of the task-specific objective functions (Chen et al., 2017).

Fine-tuning Fine-tuning has been used successfully to transfer between similar tasks, e.g. in QA (Min et al., 2017), for distantly supervised sentiment analysis (Severyn and Moschitti, 2015), or MT domains (Sennrich et al., 2015) but has been shown to fail between unrelated ones (Mou et al., 2016). Dai and Le (2015) also fine-tune a language model, but overfit with 10k labeled examples and require millions of in-domain documents for good performance. In contrast, ULMFiT leverages general-domain pretraining and novel fine-tuning techniques to prevent overfitting even with only 100 labeled examples and achieves state-of-the-art results also on small datasets.

3 Universal Language Model Fine-tuning

We are interested in the most general *inductive* transfer learning setting for NLP (Pan and Yang, 2010): Given a static source task $\mathcal{T}_S$ and *any* target task $\mathcal{T}_T$ with $\mathcal{T}_S \neq \mathcal{T}_T$, we would like to improve performance on $\mathcal{T}_T$. Language modeling can be seen as the ideal source task and a counterpart of ImageNet for NLP: It captures many facets of language relevant for downstream tasks, such as long-term dependencies (Linzen et al., 2016), hierarchical relations (Gulordava et al., 2018), and sentiment (Radford et al., 2017). In contrast to tasks like MT (McCann et al., 2017) and entailment (Conneau et al., 2017), it provides data in near-unlimited quantities for most domains and languages. Additionally, a pretrained LM can be easily adapted to the idiosyncrasies of a target

[2] A hypercolumn at a pixel in CV is the vector of activations of all CNN units above that pixel. In analogy, a hypercolumn for a word or sentence in NLP is the concatenation of embeddings at different layers in a pretrained model.



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

Figure 1: ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('*Discr*') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, '*Discr*', and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

task, which we show significantly improves performance (see Section 5). Moreover, language modeling already is a key component of existing tasks such as MT and dialogue modeling. Formally, language modeling induces a hypothesis space $\mathcal{H}$ that should be useful for many other NLP tasks (Vapnik and Kotz, 1982; Baxter, 2000).

We propose Universal Language Model Fine-tuning (ULMFiT), which pretrains a language model (LM) on a large general-domain corpus and fine-tunes it on the target task using novel techniques. The method is *universal* in the sense that it meets these practical criteria: 1) It works across tasks varying in document size, number, and label type; 2) it uses a single architecture and training process; 3) it requires no custom feature engineering or preprocessing; and 4) it does not require additional in-domain documents or labels.

In our experiments, we use the state-of-the-art language model AWD-LSTM (Merity et al., 2017a), a regular LSTM (with no attention, short-cut connections, or other sophisticated additions) with various tuned dropout hyperparameters. Analogous to CV, we expect that downstream performance can be improved by using higher-performance language models in the future.

ULMFiT consists of the following steps, which we show in Figure 1: a) General-domain LM pretraining (§3.1); b) target task LM fine-tuning (§3.2); and c) target task classifier fine-tuning (§3.3). We discuss these in the following sections.

3.1 General-domain LM pretraining

An ImageNet-like corpus for language should be large and capture general properties of language. We pretrain the language model on Wikitext-103 (Merity et al., 2017b) consisting of 28,595 preprocessed Wikipedia articles and 103 million words. Pretraining is most beneficial for tasks with small datasets and enables generalization even with 100 labeled examples. We leave the exploration of more diverse pretraining corpora to future work, but expect that they would boost performance. While this stage is the most expensive, it only needs to be performed once and improves performance and convergence of downstream models.

3.2 Target task LM fine-tuning

No matter how diverse the general-domain data used for pretraining is, the data of the target task will likely come from a different distribution. We thus fine-tune the LM on data of the target task. Given a pretrained general-domain LM, this stage converges faster as it only needs to adapt to the idiosyncrasies of the target data, and it allows us to train a robust LM even for small datasets. We propose *discriminative fine-tuning* and *slanted triangular learning rates* for fine-tuning the LM, which we introduce in the following.

Discriminative fine-tuning As different layers capture *different types of information* (Yosinski et al., 2014), they should be fine-tuned to *different extents*. To this end, we propose a novel fine-

tuning method, *discriminative fine-tuning*[3].

Instead of using the same learning rate for *all* layers of the model, discriminative fine-tuning allows us to tune *each* layer with different learning rates. For context, the regular stochastic gradient descent (SGD) update of a model's parameters θ at time step t looks like the following (Ruder, 2016):

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta) \quad (1)$$

where η is the learning rate and $\nabla_\theta J(\theta)$ is the gradient with regard to the model's objective function. For discriminative fine-tuning, we split the parameters θ into $\{\theta^1, \ldots, \theta^L\}$ where θ^l contains the parameters of the model at the l-th layer and L is the number of layers of the model. Similarly, we obtain $\{\eta^1, \ldots, \eta^L\}$ where η^l is the learning rate of the l-th layer.

The SGD update with discriminative fine-tuning is then the following:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta) \quad (2)$$

We empirically found it to work well to first choose the learning rate η^L of the last layer by fine-tuning only the last layer and using $\eta^{l-1} = \eta^l / 2.6$ as the learning rate for lower layers.

Slanted triangular learning rates For adapting its parameters to task-specific features, we would like the model to quickly converge to a suitable region of the parameter space in the beginning of training and then refine its parameters. Using the same learning rate (LR) or an annealed learning rate throughout training is not the best way to achieve this behaviour. Instead, we propose *slanted triangular learning rates* (STLR), which first linearly increases the learning rate and then linearly decays it according to the following update schedule, which can be seen in Figure 2:

$$\begin{aligned} cut &= \lfloor T \cdot cut_frac \rfloor \\ p &= \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (ratio - 1)}, & \text{otherwise} \end{cases} \\ \eta_t &= \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio} \end{aligned} \quad (3)$$

where T is the number of training iterations[4], cut_frac is the fraction of iterations we increase the LR, cut is the iteration when we switch from increasing to decreasing the LR, p is the fraction of the number of iterations we have increased or will decrease the LR respectively, $ratio$ specifies how much smaller the lowest LR is from the maximum LR η_{max}, and η_t is the learning rate at iteration t. We generally use $cut_frac = 0.1$, $ratio = 32$ and $\eta_{max} = 0.01$.

STLR modifies triangular learning rates (Smith, 2017) with a short increase and a long decay period, which we found key for good performance.[5] In Section 5, we compare against aggressive cosine annealing, a similar schedule that has recently been used to achieve state-of-the-art performance in CV (Loshchilov and Hutter, 2017).[6]

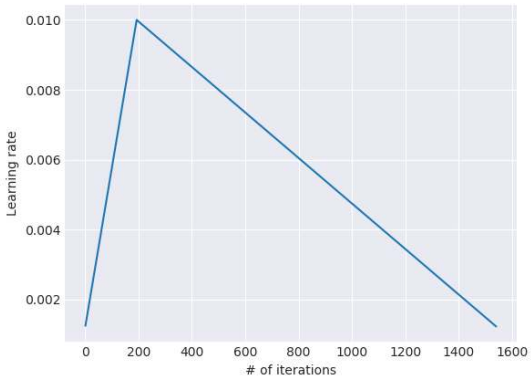


Figure 2: The slanted triangular learning rate schedule used for ULMFiT as a function of the number of training iterations.

3.3 Target task classifier fine-tuning

Finally, for fine-tuning the classifier, we augment the pretrained language model with two additional linear blocks. Following standard practice for CV classifiers, each block uses batch normalization (Ioffe and Szegedy, 2015) and dropout, with ReLU activations for the intermediate layer and a softmax activation that outputs a probability distribution over target classes at the last layer. Note that the parameters in these task-specific classifier layers are the only ones that are learned from scratch. The first linear layer takes as the input the pooled last hidden layer states.

Concat pooling The signal in text classification tasks is often contained in a few words, which may

[3] An unrelated method of the same name exists for deep Boltzmann machines (Salakhutdinov and Hinton, 2009).

[4] In other words, the number of epochs times the number of updates per epoch.

[5] We also credit personal communication with the author.

[6] While Loshchilov and Hutter (2017) use multiple annealing cycles, we generally found one cycle to work best.

occur anywhere in the document. As input documents can consist of hundreds of words, information may get lost if we only consider the last hidden state of the model. For this reason, we concatenate the hidden state at the last time step $\mathbf{h}_T$ of the document with both the max-pooled and the mean-pooled representation of the hidden states over as many time steps as fit in GPU memory $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$:

$$\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})] \quad (4)$$

where [] is concatenation.

Fine-tuning the target classifier is the most critical part of the transfer learning method. Overly aggressive fine-tuning will cause catastrophic forgetting, eliminating the benefit of the information captured through language modeling; too cautious fine-tuning will lead to slow convergence (and resultant overfitting). Besides discriminative fine-tuning and triangular learning rates, we propose *gradual unfreezing* for fine-tuning the classifier.

Gradual unfreezing Rather than fine-tuning all layers at once, which risks catastrophic forgetting, we propose to gradually unfreeze the model starting from the last layer as this contains the *least general* knowledge (Yosinski et al., 2014): We first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We then unfreeze the next lower frozen layer and repeat, until we fine-tune all layers until convergence at the last iteration. This is similar to '*chain-thaw*' (Felbo et al., 2017), except that we add a layer at a time to the set of 'thawed' layers, rather than only training a single layer at a time.

While discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing all are beneficial on their own, we show in Section 5 that they complement each other and enable our method to perform well across diverse datasets.

BPTT for Text Classification (BPT3C) Language models are trained with backpropagation through time (BPTT) to enable gradient propagation for large input sequences. In order to make fine-tuning a classifier for large documents feasible, we propose BPTT for Text Classification (BPT3C): We divide the document into fixed-length batches of size b. At the beginning of each batch, the model is initialized with the final state of the previous batch; we keep track of the hidden states for mean and max-pooling; gradients are back-propagated to the batches whose hidden states contributed to the final prediction. In practice, we use variable length backpropagation sequences (Merity et al., 2017a).

Dataset	Type	# classes	# examples
TREC-6	Question	6	5.5k
IMDb	Sentiment	2	25k
Yelp-bi	Sentiment	2	560k
Yelp-full	Sentiment	5	650k
AG	Topic	4	120k
DBpedia	Topic	14	560k

Table 1: Text classification datasets and tasks with number of classes and training examples.

Bidirectional language model Similar to existing work (Peters et al., 2017, 2018), we are not limited to fine-tuning a unidirectional language model. For all our experiments, we pretrain both a forward and a backward LM. We fine-tune a classifier for each LM independently using BPT3C and average the classifier predictions.

4 Experiments

While our approach is equally applicable to sequence labeling tasks, we focus on text classification tasks in this work due to their important real-world applications.

4.1 Experimental setup

Datasets and tasks We evaluate our method on six widely-studied datasets, with varying numbers of documents and varying document length, used by state-of-the-art text classification and transfer learning approaches (Johnson and Zhang, 2017; McCann et al., 2017) as instances of three common text classification tasks: sentiment analysis, question classification, and topic classification. We show the statistics for each dataset and task in Table 1.

Sentiment Analysis For sentiment analysis, we evaluate our approach on the binary movie review IMDb dataset (Maas et al., 2011) and on the binary and five-class version of the Yelp review dataset compiled by Zhang et al. (2015).

Question Classification We use the six-class version of the small TREC dataset (Voorhees and Tice, 1999) dataset of open-domain, fact-based questions divided into broad semantic categories.

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Table 3: Test error rates (%) on text classification datasets used by Johnson and Zhang (2017).

Topic classification For topic classification, we evaluate on the large-scale AG news and DBpedia ontology datasets created by Zhang et al. (2015).

Pre-processing We use the same pre-processing as in earlier work (Johnson and Zhang, 2017; McCann et al., 2017). In addition, to allow the language model to capture aspects that might be relevant for classification, we add special tokens for upper-case words, elongation, and repetition.

Hyperparameters We are interested in a model that performs robustly across a diverse set of tasks. To this end, if not mentioned otherwise, we use the same set of hyperparameters across tasks, which we tune on the IMDb validation set. We use the AWD-LSTM language model (Merity et al., 2017a) with an embedding size of 400, 3 layers, 1150 hidden activations per layer, and a BPTT batch size of 70. We apply dropout of 0.4 to layers, 0.3 to RNN layers, 0.4 to input embedding layers, 0.05 to embedding layers, and weight dropout of 0.5 to the RNN hidden-to-hidden matrix. The classifier has a hidden layer of size 50. We use Adam with $\beta_1 = 0.7$ instead of the default $\beta_1 = 0.9$ and $\beta_2 = 0.99$, similar to (Dozat and Manning, 2017). We use a batch size of 64, a base learning rate of 0.004 and 0.01 for fine-tuning the LM and the classifier respectively, and tune the number of epochs on the validation set of each task[7]. We otherwise use the same practices used in (Merity et al., 2017a).

[7]On small datasets such as TREC-6, we fine-tune the LM only for 15 epochs without overfitting, while we can fine-tune longer on larger datasets. We found 50 epochs to be a good default for fine-tuning the classifier.

Baselines and comparison models For each task, we compare against the current state-of-the-art. For the IMDb and TREC-6 datasets, we compare against CoVe (McCann et al., 2017), a state-of-the-art transfer learning method for NLP. For the AG, Yelp, and DBpedia datasets, we compare against the state-of-the-art text categorization method by Johnson and Zhang (2017).

4.2 Results

For consistency, we report all results as error rates (lower is better). We show the test error rates on the IMDb and TREC-6 datasets used by McCann et al. (2017) in Table 2. Our method outperforms both CoVe, a state-of-the-art transfer learning method based on hypercolumns, as well as the state-of-the-art on both datasets. On IMDb, we reduce the error dramatically by 43.9% and 22% with regard to CoVe and the state-of-the-art respectively. This is promising as the existing state-of-the-art requires complex architectures (Peters et al., 2018), multiple forms of attention (McCann et al., 2017) and sophisticated embedding schemes (Johnson and Zhang, 2016), while our method employs a regular LSTM with dropout. We note that the language model fine-tuning approach of Dai and Le (2015) only achieves an error of 7.64 vs. 4.6 for our method on IMDb, demonstrating the benefit of transferring knowledge from a large ImageNet-like corpus using our fine-tuning techniques. IMDb in particular is reflective of real-world datasets: Its documents are generally a few

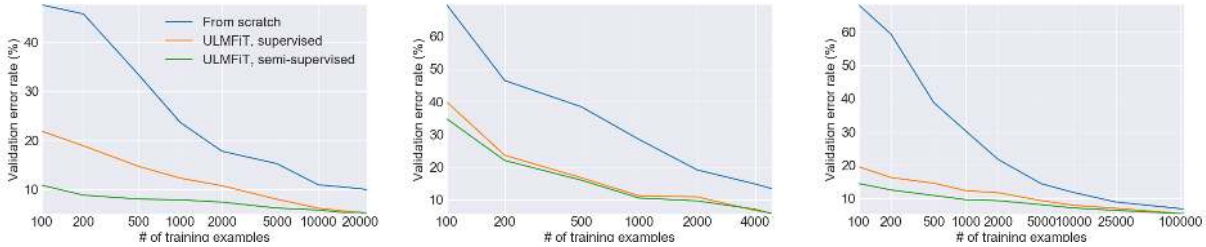


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

paragraphs long—similar to emails (e.g for legal discovery) and online comments (e.g for community management); and sentiment analysis is similar to many commercial applications, e.g. product response tracking and support email routing.

On TREC-6, our improvement—similar as the improvements of state-of-the-art approaches—is not statistically significant, due to the small size of the 500-examples test set. Nevertheless, the competitive performance on TREC-6 demonstrates that our model performs well across different dataset sizes and can deal with examples that range from single sentences—in the case of TREC-6—to several paragraphs for IMDb. Note that despite pretraining on more than two orders of magnitude less data than the 7 million sentence pairs used by McCann et al. (2017), we consistently outperform their approach on both datasets.

We show the test error rates on the larger AG, DBpedia, Yelp-bi, and Yelp-full datasets in Table 3. Our method again outperforms the state-of-the-art significantly. On AG, we observe a similarly dramatic error reduction by 23.7% compared to the state-of-the-art. On DBpedia, Yelp-bi, and Yelp-full, we reduce the error by 4.8%, 18.2%, 2.0% respectively.

5 Analysis

In order to assess the impact of each contribution, we perform a series of analyses and ablations. We run experiments on three corpora, IMDb, TREC-6, and AG that are representative of different tasks, genres, and sizes. For all experiments, we split off 10% of the training set and report error rates on this validation set with unidirectional LMs. We fine-tune the classifier for 50 epochs and train all methods but ULMFiT with early stopping.

Low-shot learning One of the main benefits of transfer learning is being able to train a model for a task with a small number of labels. We evaluate ULMFiT on different numbers of labeled examples in two settings: only labeled examples are used for LM fine-tuning ('*supervised*'); and all task data is available and can be used to fine-tune the LM ('*semi-supervised*'). We compare ULMFiT to training from scratch—which is necessary for hypercolumn-based approaches. We split off balanced fractions of the training data, keep the validation set fixed, and use the same hyperparameters as before. We show the results in Figure 3.

On IMDb and AG, supervised ULMFiT with only 100 labeled examples matches the performance of training from scratch with $10\times$ and $20\times$ more data respectively, clearly demonstrating the benefit of general-domain LM pretraining. If we allow ULMFiT to also utilize unlabeled examples (50k for IMDb, 100k for AG), at 100 labeled examples, we match the performance of training from scratch with $50\times$ and $100\times$ more data on AG and IMDb respectively. On TREC-6, ULMFiT significantly improves upon training from scratch; as examples are shorter and fewer, supervised and semi-supervised ULMFiT achieve similar results.

Pretraining	IMDb	TREC-6	AG
Without pretraining	5.63	10.67	5.52
With pretraining	**5.00**	**5.69**	**5.38**

Table 4: Validation error rates for ULMFiT with and without pretraining.

Impact of pretraining We compare using no pretraining with pretraining on WikiText-103 (Merity et al., 2017b) in Table 4. Pretraining is most useful for small and medium-sized datasets, which are most common in commercial applications. However, even for large datasets, pretraining improves performance.

LM	IMDb	TREC-6	AG
Vanilla LM	5.98	7.41	5.76
AWD-LSTM LM	**5.00**	**5.69**	**5.38**

Table 5: Validation error rates for ULMFiT with a vanilla LM and the AWD-LSTM LM.

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

Impact of LM quality In order to gauge the importance of choosing an appropriate LM, we compare a vanilla LM with the same hyperparameters without any dropout[8] with the AWD-LSTM LM with tuned dropout parameters in Table 5. Using our fine-tuning techniques, even a regular LM reaches surprisingly good performance on the larger datasets. On the smaller TREC-6, a vanilla LM without dropout runs the risk of overfitting, which decreases performance.

Impact of LM fine-tuning We compare no fine-tuning against fine-tuning the full model (Erhan et al., 2010) ('*Full*'), the most commonly used fine-tuning method, with and without discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*') in Table 6. Fine-tuning the LM is most beneficial for larger datasets. '*Discr*' and '*Stlr*' improve performance across all three datasets and are necessary on the smaller TREC-6, where regular fine-tuning is not beneficial.

Impact of classifier fine-tuning We compare training from scratch, fine-tuning the full model ('*Full*'), only fine-tuning the last layer ('*Last*') (Donahue et al., 2014), '*Chain-thaw*' (Felbo et al., 2017), and gradual unfreezing ('*Freez*'). We furthermore assess the importance of discriminative fine-tuning ('*Discr*') and slanted triangular learning rates ('*Stlr*'). We compare the latter to an alternative, aggressive cosine annealing schedule ('*Cos*') (Loshchilov and Hutter, 2017). We use a learning rate $\eta^L = 0.01$ for '*Discr*', learning rates of 0.001 and 0.0001 for the last and all other layers respectively for '*Chain-thaw*' as in (Felbo et al., 2017), and a learning rate of 0.001 otherwise. We show the results in Table 7.

[8]To avoid overfitting, we only train the vanilla LM classifier for 5 epochs and keep dropout of 0.4 in the classifier.

Classifier fine-tuning	IMDb	TREC-6	AG
From scratch	9.93	13.36	6.81
Full	6.87	6.86	5.81
Full + discr	4.57	6.21	5.62
Last	6.49	16.09	8.38
Chain-thaw	5.39	6.71	5.90
Freez	6.37	6.86	5.81
Freez + discr	5.39	5.86	6.04
Freez + stlr	5.04	6.02	5.35
Freez + cos	5.70	6.38	**5.29**
Freez + discr + stlr	**5.00**	**5.69**	5.38

Table 7: Validation error rates for ULMFiT with different methods to fine-tune the classifier.

Fine-tuning the classifier significantly improves over training from scratch, particularly on the small TREC-6. '*Last*', the standard fine-tuning method in CV, severely underfits and is never able to lower the training error to 0. '*Chain-thaw*' achieves competitive performance on the smaller datasets, but is outperformed significantly on the large AG. '*Freez*' provides similar performance as '*Full*'. '*Discr*' consistently boosts the performance of '*Full*' and '*Freez*', except for the large AG. Cosine annealing is competitive with slanted triangular learning rates on large data, but under-performs on smaller datasets. Finally, full ULMFiT classifier fine-tuning (bottom row) achieves the best performance on IMDB and TREC-6 and competitive performance on AG. Importantly, ULMFiT is the only method that shows excellent performance across the board—and is therefore the only *universal* method.

Classifier fine-tuning behavior While our results demonstrate that *how* we fine-tune the classifier makes a significant difference, fine-tuning for inductive transfer is currently under-explored in NLP as it mostly has been thought to be unhelpful (Mou et al., 2016). To better understand the fine-tuning behavior of our model, we compare the validation error of the classifier fine-tuned with ULMFiT and '*Full*' during training in Figure 4.

On all datasets, fine-tuning the full model leads to the lowest error comparatively early in training, e.g. already after the first epoch on IMDb.

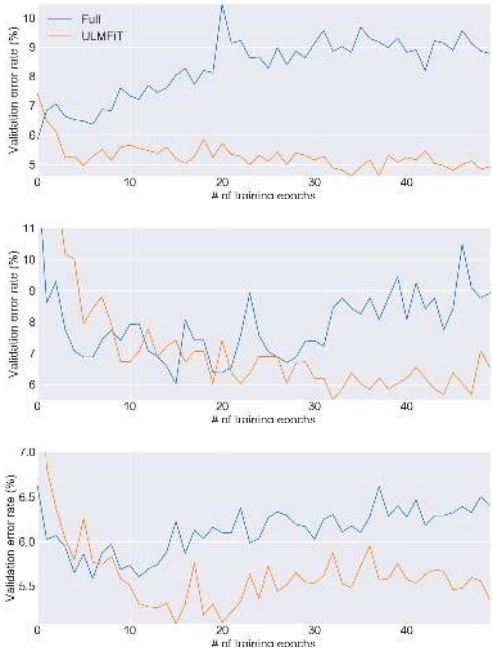


Figure 4: Validation error rate curves for fine-tuning the classifier with ULMFiT and '*Full*' on IMDb, TREC-6, and AG (top to bottom).

The error then increases as the model starts to overfit and knowledge captured through pretraining is lost. In contrast, ULMFiT is more stable and suffers from no such catastrophic forgetting; performance remains similar or improves until late epochs, which shows the positive effect of the learning rate schedule.

Impact of bidirectionality At the cost of training a second model, ensembling the predictions of a forward and backwards LM-classifier brings a performance boost of around 0.5–0.7. On IMDb we lower the test error from 5.30 of a single model to 4.58 for the bidirectional model.

6 Discussion and future directions

While we have shown that ULMFiT can achieve state-of-the-art performance on widely used text classification tasks, we believe that language model fine-tuning will be particularly useful in the following settings compared to existing transfer learning approaches (Conneau et al., 2017; McCann et al., 2017; Peters et al., 2018): a) NLP for non-English languages, where training data for supervised pretraining tasks is scarce; b) new NLP tasks where no state-of-the-art architecture exists; and c) tasks with limited amounts of labeled data (and some amounts of unlabeled data).

Given that transfer learning and particularly fine-tuning for NLP is under-explored, many future directions are possible. One possible direction is to improve language model pretraining and fine-tuning and make them more scalable: for ImageNet, predicting far fewer classes only incurs a small performance drop (Huh et al., 2016), while recent work shows that an alignment between source and target task label sets is important (Mahajan et al., 2018)—focusing on predicting a subset of words such as the most frequent ones might retain most of the performance while speeding up training. Language modeling can also be augmented with additional tasks in a multi-task learning fashion (Caruana, 1993) or enriched with additional supervision, e.g. syntax-sensitive dependencies (Linzen et al., 2016) to create a model that is more general or better suited for certain downstream tasks, ideally in a weakly-supervised manner to retain its universal properties.

Another direction is to apply the method to novel tasks and models. While an extension to sequence labeling is straightforward, other tasks with more complex interactions such as entailment or question answering may require novel ways to pretrain and fine-tune. Finally, while we have provided a series of analyses and ablations, more studies are required to better understand what knowledge a pretrained language model captures, how this changes during fine-tuning, and what information different tasks require.

7 Conclusion

We have proposed ULMFiT, an effective and extremely sample-efficient transfer learning method that can be applied to any NLP task. We have also proposed several novel fine-tuning techniques that in conjunction prevent catastrophic forgetting and enable robust learning across a diverse range of tasks. Our method significantly outperformed existing transfer learning techniques and the state-of-the-art on six representative text classification tasks. We hope that our results will catalyze new developments in transfer learning for NLP.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289.

References

Jonathan Baxter. 2000. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research* 12:149–198.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics* 45(1):440. https://doi.org/10.1109/IRPS.2011.5784441.

Cornelia Caragea, Nathan McNeese, Anuj Jaiswal, Greg Traylor, Hyun-Woo Kim, Prasenjit Mitra, Dinghao Wu, Andrea H Tapia, Lee Giles, Bernard J Jansen, et al. 2011. Classifying text messages for the haiti earthquake. In *Proceedings of the 8th international conference on information systems for crisis response and management (ISCRAM2011)*. Citeseer.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks pages 1–10.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2012. Detecting automation of twitter accounts: Are you a human, bot, or cyborg? *IEEE Transactions on Dependable and Secure Computing* 9(6):811–824.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised Sequence Learning. *Advances in Neural Information Processing Systems (NIPS '15)* http://arxiv.org/abs/1511.01432.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*. pages 647–655.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of ICLR 2017*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11(Feb):625–660.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL-HLT 2018*.

Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 447–456.

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*.

Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten. 2017. Densely Connected Convolutional Networks. In *Proceedings of CVPR 2017*.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. 2016. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614* .

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*. pages 448–456.

Nitin Jindal and Bing Liu. 2007. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*. ACM, pages 1189–1190.

Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *International Conference on Machine Learning*. pages 526–534.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 562–570.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368* .

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Proceedings of AAAI 2018*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015a. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3431–3440.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015b. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the 32nd International Conference on Machine learning (ICML '15)*. volume 37.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proceedings of the Internal Conference on Learning Representations 2017*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the Limits of Weakly Supervised Pretraining .

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017a. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182* .

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017b. Pointer Sentinel Mixture Models. In *Proceedings of the International Conference on Learning Representations 2017*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* .

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How Transferable are Neural Networks in NLP Applications? *Proceedings of 2016 Conference on Empirical Methods in Natural Language Processing* .

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

EWT Ngai, Yong Hu, YH Wong, Yijun Chen, and Xin Sun. 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems* 50(3):559–569.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22(10):1345–1359.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of ACL 2017*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL 2018*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444* .

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of ACL 2017*.

Herbert L Roitblat, Anne Kershaw, and Patrick Oot. 2010. Document categorization in legal electronic discovery: computer classification vs. manual review. *Journal of the Association for Information Science and Technology* 61(1):70–80.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* .

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep boltzmann machines. In *Artificial Intelligence and Statistics*. pages 448–455.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Aliaksei Severyn and Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* pages 464–469.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 806–813.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, pages 464–472.

Vladimir Naumovich Vapnik and Samuel Kotz. 1982. *Estimation of dependences based on empirical data*, volume 40. Springer-Verlag New York.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*. volume 1999, page 82.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*. pages 3320–3328.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. pages 649–657.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016*.