Does an LSTM forget more than a CNN? An empirical study of catastrophic forgetting in NLP

Gaurav Arora **Afshin Rahimi** **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne
gaurava@student.unimelb.edu.au
{arahimi,tbaldwin}@unimelb.edu.au

Abstract

Catastrophic forgetting — whereby a model trained on one task is fine-tuned on a second, and in doing so, suffers a "catastrophic" drop in performance over the first task — is a hurdle in the development of better transfer learning techniques. Despite impressive progress in reducing catastrophic forgetting, we have limited understanding of how different architectures and hyper-parameters affect forgetting in a network. In this paper, we aim to understand factors which cause forgetting during sequential training. Our primary finding is that CNNs forget less than LSTMs. We show that max-pooling is the underlying operation which helps CNNs alleviate forgetting compared to LSTMs. We also found that curriculum learning (Bengio et al., 2009), placing a hard task towards the end of task sequence, reduces forgetting. We analysed the effect of fine-tuning contextual embeddings on catastrophic forgetting, and found that using fixed word embeddings is preferable to fine-tuning.[1]

1 Introduction

Transfer learning — transferring knowledge from a source task to a target task — has become an essential technique in both computer vision and NLP. Earlier attempts at transfer learning were limited in their applicability (Mou et al., 2016), as the transfer only worked for very similar tasks: if the source and target tasks were not very similar, training on the target task resulted in catastrophic forgetting (Ratcliff, 1990; McCloskey and Cohen, 1989), whereby the neural network abruptly forgets previously-acquired knowledge during training on a new task, limiting inductive transfer. ULMFit (Howard and Ruder, 2018) developed specialised techniques to reduce forgetting during the fine-tuning process, resulting in successful transfer learning. A general finding of this work was that uncovering underlying causes of catastrophic forgetting can result in improved architectures for transfer learning.

Previous studies found that using dropout (Goodfellow et al., 2014) and sharp activation functions (French, 1991) help reduce catastrophic forgetting. Sharp activation functions effectively distribute each task to different parts of the network. It is unknown if increasing the capacity of the network will also have a similar effect. Another study found that using a max operation (Srivastava et al., 2013) in the network reduces forgetting. There hasn't been a study comparing forgetting for different architectures to test if networks with the max operation have less forgetting. Task complexity (Nguyen et al., 2019) is positively correlated with total error observed, but it is unknown how we should arrange tasks in sequence to reduce forgetting. We conduct an empirical study to understand how factors like network architecture and capacity affect forgetting.

In this work, we design experiments to empirically address the following research questions:

RQ1: Do some neural architectures forget more than others?

RQ2 Should we fine-tune pre-trained embeddings in a continual learning setup?

RQ3 Do networks with more capacity forget less?

RQ4 Do networks forget more during/after training over a difficult task?

Our experimental setup consists of studying forgetting for various neural architectures and hyper-parameter configurations in a continual learning setup. We train the network without access to data from the previous tasks, and measure how much of the knowledge learned in previous tasks is forgotten. After performing initial experiments, we

[1] All code associated with this paper is available at https://github.com/gauravaror/catastrophic_forgetting

conduct further experiments to understand the underlying reason for the differences in forgetting.

We found that CNNs forget less than LSTMs, because of max pooling. Max-pooling decreases forgetting as the gradient doesn't update all the shared parameters. Further, adding contextual word embeddings such as ELMo (Peters et al., 2018a) with either an LSTM or a CNN as the top layer, reduces the forgetting for both architectures. Surprisingly, the LSTM forgets less when the ELMo embeddings are frozen, and fine-tuning performs worse than randomly initialised embeddings in a continual learning setup. We also found that, contrary to common wisdom, more network capacity doesn't always result in less forgetting. For CNNs, sequence forgetting increases as we increase the number of layers, whereas for the dimensionality of hidden layers, the degree of forgetting depends on the task sequence: the choice of which task to train first has more impact on forgetting than the number of hidden units in the network, and placing difficult tasks towards the end of the task sequence reduces overall forgetting.

2 Background

2.1 Catastrophic forgetting

Our work is similar to early work on catastrophic forgetting (Ratcliff, 1990; McCloskey and Cohen, 1989), which studied factors affecting forgetting like the width of the network or amount of training. The amount of new learning was found to be directly proportional to the amount of forgetting in previous tasks. They also found that lowering the learning rate decreases forgetting, but impairs the ability of the network to learn. Recent empirical work (Goodfellow et al., 2014) studied the effect of the activation function and different training algorithms. They found that training with dropout is always better, and the choice of activation is task-dependent and should be cross-validated.

Both earlier empirical studies focused on only two-task sequences, whereas we use four-task sequences, based on Nguyen et al. (2019) who studied the effect of total sequence complexity and sequential heterogeneity. They found that error rates do not correlate with the sequential heterogeneity of tasks.

Rebuffi et al. (2017) and Li and Hoiem (2018) found that training on a subsequent task doesn't update the classification layer of the previous task, and only updates the encoder layer, increasing catastrophic forgetting. Knowledge distillation loss (Hinton et al., 2015) is a commonly used technique to avoid dramatic changes in the encoder layer, while adapting the classification layer for the new task.

Yogatama et al. (2019) found catastrophic forgetting while fine-tuning ELMo and BERT (Devlin et al., 2019) embeddings, similar to our findings. They also found that sampling examples from a different task (with uniform probability) enables a network to learn all tasks reasonably well; this requires access to all task simultaneously, which is different from our setup. We consider whether to fine-tune embeddings or not, which is similar to the question posed by Peters et al. (2019), who focused on various types of task. In contrast, we address the same question in a continual learning setup.

2.2 Transfer Learning

ULMFit (Howard and Ruder, 2018) was an effort to enable transfer learning in a pre-trained LSTM network. The authors utilised specialised techniques such as layer-wise fine-tuning, concat pooling (concatenation of final hidden state), and max and mean pooling of all hidden states to alleviate catastrophic forgetting during fine-tuning. In this work, we argue and empirically support the use of max pooling, as opposed to using only average pooling, as a means to reduce catastrophic forgetting. It would be interesting to further study the individual architectural design choices that enable successful transfer learning, which we leave to future work.

2.3 Evaluation metrics for Catastrophic forgetting

GEM (Lopez-Paz et al., 2017) proposed Average, Backward, and Forward Transfer to measure catastrophic forgetting. Backward Transfer measures the influence task t has on the previous task $k < t$, and Forward Transfer measures the influence on future task $k > t$. Since we are only concerned with forgetting in the network, we use Backward Transfer with a slight modification to measure catastrophic forgetting in a task sequence.

The amount of absolute drop in task performance is not a good measure of forgetting because tasks have different state-of-the-art (SOTA) performance and difficulty level (e.g. majority class performance). The forgetting ratio (Serra et al.,

2018) is a normalised measure of forgetting across multiple tasks which we also adapt in this work.

3 Method

We study catastrophic forgetting by training tasks sequentially. During sequential task training, networks suffer from forgetting knowledge acquired in previous tasks because of overfitting to new tasks, and also lack of access to the training data of the old tasks. Our setup is very similar to fine-tuning in transfer learning. Various tasks and task sequences used in our study are described in Section 4. Task sequences are trained using neural architectures with fixed hyper-parameters, as described in Section 6. We performed experiments to find how forgetting changes for different architectures (Section 7), ways to use embeddings (Section 8), network configurations (Section 9), and task sequences (Section 10). We compare the amount of forgetting of various architectural design choices using the evaluation metric proposed in Section 5.

4 Tasks

We selected four text classification tasks of different nature, each targeting different language learning tasks for English.

- **Stanford Sentiment Treebank ("SST"):** fine-grained sentiment classification over five classes (Socher et al., 2013).
- **Subjectivity ("SUBJ"):** binary classification of Subjectivity vs. Objectivity in IMDB reviews (Pang and Lee, 2004).
- **TREC Question classification ("TREC"):** coarse-grained classification of questions, based on 6 classes (Voorhees and Tice, 1999).
- **Corpus of Linguistic Acceptability ("CoLA"):** prediction of whether a sentence is grammatical or not (Warstadt et al., 2018).

Table 1 contains state-of-the-art (SOTA), majority class voting, and single-task performance using a CNN for all four tasks.

We consider a task difficult for our setup if we cannot attain performance close to SOTA with a simple architecture like an LSTM or CNN. SST is the most challenging task in our setup: achieving SOTA performance requires large pre-trained contextual embeddings like ELMo (Peters et al., 2018a). Socher et al. (2013) proposed recursive neural networks for SST based on an explicit constituency parse tree, and results for standard LSTMs are well below SOTA. CoLA is a moderately difficult task as it also requires specialised techniques to perform reasonably well. For TREC and Subjectivity, on the other hand, it is possible to reach performance close to SOTA with simple architectures. We intentionally selected tasks of varying difficulty to see if forgetting increases with more complicated tasks.

4.1 Task Sequence

We formed various task sequences with length four using the tasks detailed in Table 1. We used all 24 task sequences possible. A selection of task sequences is listed in Table 6, wherein the code name indicates the order of the tasks during training (e.g. "TREC_SUBJ_CoLA_SST" = train on TREC first, then SUBJ, CoLA and SST).

5 Evaluation

We used accuracy as our metric for evaluation, except for CoLA where we used Mathews correlation (Matthews, 1975) as the dataset is unbalanced. All results are averaged over five runs with different random seeds.

After training a sequence, we calculate the performance score for each component task (over held-out test data). Because absolute metrics are not comparable between tasks, we normalise the raw performance score for each task to get a roughly uniform metric, disregarding task difficulty. Further, we use normalised performance scores to calculate forgetting for each task and the whole task sequence, as detailed below.

5.1 Normalisation

Direct comparison of forgetting between TREC and SST, e.g., is not ideal, as the absolute difference in accuracy could be up to 40%. Normalisation enables fairer comparison of forgetting, as it incorporates a measure of task difficulty based on SOTA and majority class performance. This normalisation is similar to the forgetting ratio proposed by Serra et al. (2018).

We normalise performance metric based on: (a) SOTA for the task $\mathrm{PER}_{\mathrm{SOTA}}$; and (b) majority class performance $\mathrm{PER}_{\mathrm{MAJ}}$. $\mathrm{PER}_{i,j}$ refers to performance measured for the task at position i after training the task at position j (where $i \leq j$). $P_{i,j}$

Tasks	SOTA	Majority Class	CNN	#Training Instances	#Classes
TREC	0.98 Cer et al. (2018)	0.19	0.91	5452	6
SUBJ	0.96 Cer et al. (2018)	0.50	0.92	9000	2
CoLA	0.34 Warstadt et al. (2018)	0.00	0.25	8551	2
SST	0.55 Peters et al. (2018a)	0.25	0.38	8544	5

Table 1: The tasks targeted in this work, with state of the art (SOTA) performance, majority class performance, and performance when trained individually using a single-layer CNN. We used Mathew's Correlation Coefficient (Matthews, 1975) for CoLA, and accuracy as the performance measure for all other tasks.

refers to the normalised performance of the task at position i after training the task at position j. Negative values for $P_{i,i}$ indicate accuracy is below majority classifier accuracy.

$$P_{i,j} = \frac{\text{PER}_{i,j} - \text{PER}_{\text{MAJ}}}{\text{PER}_{\text{SOTA}} - \text{PER}_{\text{MAJ}}} \quad \forall i \leq j \qquad (1)$$

5.2 Forgetting of a Sequence

We use this normalised performance to measure forgetting for an entire task sequence. Our forgetting metric is similar to Backward Transfer proposed by Lopez-Paz et al. (2017). We track forgetting of the task sequence, which is a scaled version of Backward Transfer.

Sequence forgetting (F_{Seq}) is the sum over the individual task forgetting values F_i. Individual task forgetting is the scaled performance drop for each task, indexed based on the position at which the task was trained. The difference between performance when the task was first trained and the end of the sequence, is considered to be performance drop:

$$F_i = \frac{P_{i,i} - P_{i,T}}{\left|P_{i,i}\right|} \qquad (2)$$

$$F_{Seq} = \sum_{i=1}^{i=T} F_i \qquad (3)$$

where T refers to the position of the last trained task. We also refer to F_i as F_{TASK} when TASK is trained at position i (e.g. when TREC is trained as the first task, $F_1 = F_{\text{TREC}}$). Lower forgetting is better.

6 Neural Models

Our network consists of an encoder and a classification layer. The encoder learns to extract useful features for the task automatically. The classification layer uses the encoder output to label instances, which is dependent on the task and actual label set. We use the same encoder but different classifier layers for all tasks. Our architecture is similar to the one used by Li and Hoiem (2018). The AllenNLP library (Gardner et al., 2018) was used to build our neural models. We used Mathew's Correlation as the early stopping criteria for CoLA, and loss for the other tasks. Unless otherwise stated, we train networks with the hyper-parameters listed in Table 2.

Optimiser	Adam
Learning Rate	0.001
Patience	10
Batch size	128
dropout	0.5
Embedding dimension	128
ELMo hidden size	1024

Table 2: Hyper-parameters used for training.

7 RQ1: Do LSTMs forget more than CNNs?

CNN-based architectures (Krizhevsky et al., 2012) have been widely used for transfer learning, whereas LSTM-based architectures need specialised techniques to transfer successfully (Howard and Ruder, 2018). This difference motivated us to compare forgetting between LSTMs and CNNs.

We used the standard LSTM encoder implementation from AllenNLP. The CNN encoder in AllenNLP is single-layered, which we adapted to

multi-layer with max-pooling applied after the final layer. We used a single n-gram filter with width two for the CNN. We ran experiments using LSTM and CNN encoders with all task sequences and network configurations.

7.1 Results: CNN vs. LSTM

Table 3 and Figure 1 compare the main results for forgetting between LSTMs and CNNs on task sequence TREC_SUBJ_SST_CoLA. Single-layered CNN networks forget considerably less than LSTM networks. The lowest Sequence Forgetting value of $F_{Seq} = 1.52$ for LSTMs is more than double the lowest F_{Seq} of 0.71 observed for CNNs. CNNs perform substantially better with single-layered networks, and forgetting starts increasing with higher numbers of layers. With higher numbers of CNN layers, forgetting is only slightly lower than LSTM networks. We also performed experiments with bi-directional LSTMs and observed very small-scale reductions in forgetting, which could be due to slightly better modelling of the task; because of the marginal difference in performance, we omit results for bi-directional LSTMs from the paper.

We conducted further experiments to understand what makes single-layered CNNs special in reducing forgetting. Convolution and pooling operations are two distinctive features of CNNs. We ran experiments replacing max-pooling with average pooling.

7.2 Results: max pooling vs. average pooling

Table 4 and Figure 2 compare the main results for forgetting between max pooling and average pooling on task sequence TREC_SUBJ_SST_CoLA. Replacing max pooling with average pooling resulted in a slight increase in F_{Seq}, indicating max-pooling helps in reducing forgetting.

A network with max pooling can train on different input distributions with less interference, as different sub-networks (paths created by max-pooling) can be used for each input distribution. Srivastava et al. (2013) also report less forgetting using a max operation in their proposed networks. We observe that even with average pooling, forgetting in CNNs is not as severe as in LSTMs.

8 RQ2: Should we fine-tune pre-trained embedding in continual learning setup?

Contextual embeddings like ELMo (Peters et al., 2018b) and BERT (Devlin et al., 2019) have became a standard component in recent NLP architectures. The embeddings used in these pre-trained architectures encode latent linguistic features from a large corpus, thus improving sample efficiency and generalisability of models, which could change the forgetting dynamics of the network. We used ELMo embeddings in a continual learning setup, and compared the model's forgetting in two scenarios: (a) embeddings are fine-tuned during each task's training; and (b) embeddings are fixed. Our experiments using fine-tuned and fixed ELMo embeddings are referred to as "CNN$_{Fix}$" and "CNN$_{FT}$" respectively, in the case of the CNN.

8.1 Results: fixed ELMo

Table 3 presents results with fixed ELMo. Freezing ELMo's parameters in continual learning reduces the forgetting, e.g. for a single-layered LSTM with 400 hidden dimensions, forgetting was reduced from 2.57 to 0.58, which is the least forgetting overall. The impact of fixed embeddings is similar for both LSTMs and CNNs. Surprisingly, LSTMs perform slightly better than CNN's, contrary to results when embeddings are not used. We hypothesise that this is due to the LSTM sharing a similar structure to the underlying model used by ELMo.

8.2 Results: ELMo with fine-tuning

Table 3 compares results using fixed and fine-tuned ELMo embeddings. While fixed ELMo helps the networks reduce forgetting, fine-tuning catastrophically degrades the networks' ability to retain previous knowledge. Most of the gain from using contextual embeddings is lost if we fine-tune the embeddings: our results show that fine-tuning increases forgetting from 0.58 to 2.57 in a single-layer LSTM network with 400 hidden dimensions. These results highlight the importance of specialised fine-tuning techniques like gradual unfreezing and discriminative fine-tuning in ULMFit (Howard and Ruder, 2018). Interestingly, the CNN performs better with fine-tuning, but the LSTM performs better with fixed ELMo embeddings.

#Layers	Hdim	CNN	LSTM	CNN_R	LSTM_R	CNN_{Fix}	LSTM_{Fix}	CNN_{FT}	LSTM_{FT}
1	100	**0.71**	**1.52**	**0.97**	**1.78**	0.77	0.71	1.90	2.63
1	400	0.76	2.24	0.98	2.57	**0.63**	**0.58**	**1.46**	**2.57**
2	100	1.51	2.23	1.94	1.92	1.02	0.85	2.28	2.81
2	400	1.74	2.11	1.95	2.30	1.01	0.95	2.08	2.19
3	100	2.03	2.04	2.02	2.13	1.83	1.53	2.16	2.33
3	400	2.04	1.81	2.45	1.79	1.18	1.47	2.28	2.33

Table 3: Sequence Forgetting (F_{Seq}) of TREC_SUBJ_SST_CoLA using CNN and LSTM for various network configurations. We denote the experiment with regularisation as "CNN_R", fixed ELMo as "CNN_{Fix}", and ELMo with fine-tuning as "CNN_{FT}". Similar notation is used for the LSTM, and "Hdim" denotes the dimensionality of the given hidden layer.

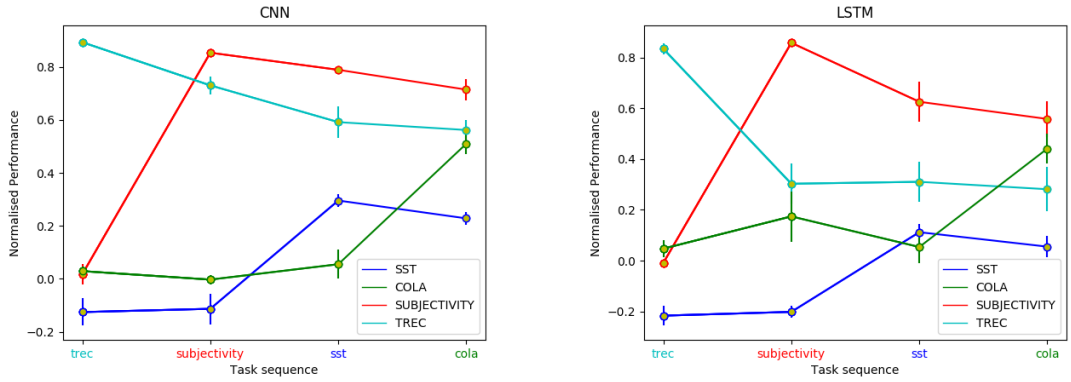

Figure 1: Performance of the LSTM and CNN on task sequence TREC_SUBJ_SST_COLA, with one layer and hidden dimensionality 100.

#Layers	Hdim	max pool	avg pool
1	100	**0.71**	0.97
1	400	0.75	0.97
1	900	0.72	**0.90**
2	100	1.66	1.61
2	400	1.72	1.83
2	900	1.58	2.18

Table 4: Sequence Forgetting (F_{Seq}) of TREC_SUBJ_SST_CoLA using max pooling and average pooling for a different configuration.

9 RQ3: Do networks with more capacity forget less?

A lot of work in the catastrophic forgetting literature has focused on freezing the weights of the network (Mallya and Lazebnik, 2018; Fernando et al., 2017). Here, we ask whether increasing the capacity of the network would encourage the network to use different sub-networks for different tasks. Another thought was that increasing capacity would drive the network to over-fit, which could further increase forgetting.

We considered neural networks up to four layers deep, with hidden dimensions of 100, 400, 900, and 1400. We trained each task sequence on sixteen different network configurations formed using four different layers and hidden dimensionalities. The hidden dimensionality refers to the number of features in the hidden state in an LSTM, or the number of output channels in a CNN.

Since networks with greater capacity are more vulnerable to over-fitting, we also studied the effect of regularising the network. We also ran all experiments using L2 regularisation by setting weight decay to 0.0001 during training. LSTM_R and CNN_R denote results for experiments with L2 regularisation.

9.1 Results: Layers

Table 3 lists results for different layers for both CNNs and LSTMs, and their regularised versions. Both CNNs and LSTMs have the least forgetting

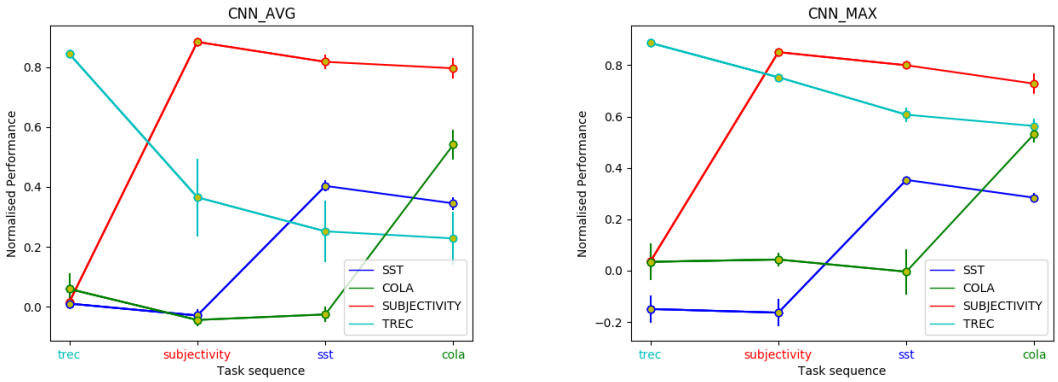

Figure 2: Performance of Max and Average pooling on task sequence: TREC_SUBJ_SST_CoLA, with Layer 1 and Hidden Dimension 100.

for the single-layered network. By increasing the number of layers in the network, forgetting also increases. For CNNs, there is a huge degradation when changing the network from one layer to two layers. This steep increase could be because we are using max-pooling only after the final layer. We might be able to reduce forgetting in a multi-layered CNN by having a pooling operation after every layer. We found that regularising using weight decay didn't help in reducing forgetting. Our finding differs that of previous work on dropout (Goodfellow et al., 2014), which found that it helps in reducing forgetting. However, the results are not comparable due to the different type of regularisation.

9.2 Results: Hidden Dimensionality

Table 5 shows forgetting for hidden dimensionalities 100 and 900 for four task sequences. We limit our analysis to single-layered CNNs without embeddings. We find that the hidden dimensionality with least forgetting is dependent on the arrangement of tasks in the task sequence. We observe that task sequences starting with TREC and CoLA have lower forgetting with $hdim = 100$ (the first and fourth task sequences in Table 5). In contrast, task sequences starting with SST and Subjectivity have less forgetting with $hdim = 900$ (the second and third task sequence in Table 5). Increasing the dimensionality reduces forgetting for some tasks, but increases forgetting for others. For task sequence CoLA_SST_TREC_SUBJ in Table 5, increasing the dimensionality from 100 to 900 increases forgetting for CoLA by 0.61, whereas forgetting for SST is reduced by 0.33. Out of the 24 tasks sequences considered, only four have $|F^{900}_{Seq} - F^{100}_{Seq}| > 0.3$.[2]

10 RQ4: Do networks forget more when training a difficult task?

From our experiments on task sequences of length four, we observed F_{Seq} varies from 0.63 to 1.81 on different task sequences. The task ordering has a substantial impact on F_{Seq}. To understand the forgetting behaviour for tasks individually, we ran an experiment with two tasks. We train two tasks sequentially, and report the forgetting observed on the first task after training the second task, averaged over five runs. We train all twelve possible configurations using four tasks.

10.1 Results: Two-task Sequence

Table 7 lists the results of forgetting on sequence lengths two with a single-layered network and hidden dimensionality of 100. We observed training a difficult task causes overall less forgetting for the previous task. Training TREC after Subjectivity results in forgetting of 0.46, whereas training hard tasks like SST only results in forgetting of 0.17. We also observe TREC suffers distinctively less forgetting by training SST and CoLA, which is also observable in our results for the four-task sequence.

To further understand why training a difficult task leads to less forgetting, we recorded the total number of epochs used in training each task. Table 8 shows the number of epochs used for the second task in training two-task sequences. TREC

[2]The superscript on F_{Seq} here indicates the dimensionality.

Code	Hdim	F_{TOTAL}	F_{TREC}	F_{CoLA}	F_{SST}	F_{SUBJ}
TREC_SUBJ_SST_CoLA	100	0.77	0.40	0.0	0.23	0.12
	900	+0.32	+0.215	0.0	+0.02	+0.08
SST_CoLA_TREC_SUBJ	100	1.36	0.12	0.58	0.65	0.0
	900	−0.41	−0.02	−0.22	−0.18	0.0
SUBJ_CoLA_SST_TREC	100	1.81	0.0	0.61	0.61	0.59
	900	−0.59	0.0	−0.35	−0.21	−0.03
CoLA_SST_TREC_SUBJ	100	0.99	0.15	0.31	0.54	0.0
	900	+0.23	−0.04	+0.61	−0.33	0.0
SST_TREC_SUBJ_COLA	100	0.93	0.19	0.0	0.61	0.12
	900	−0.10	+0.10	0.0	−0.16	−0.04

Table 5: F_{Seq} and individual task forgetting for four task sequences on a single-layered network with hidden dimensionality 100 and 900. The actual forgetting value is reported for dimensionality 100 and increment/decrement from reference value at 100 dimensionality is reported for 900 dimensionality, with red indicating an increase in forgetting and green indicating a decrease in forgetting from 100 dimensionality.

generally requires more epochs than other tasks, accounting for the large drop in performance when training TREC later in the sequence.

10.2 Results: Forgetting when training a difficult task

Table 6 lists the top and bottom task sequences based on minimum F_{Seq} across all considered dimensionalities. Results are in line with our observation from the two-task sequence, that training a difficult task causes less forgetting to tasks trained earlier. Having a difficult task like SST towards the end of a sequence reduces overall forgetting. In Table 6, all the top task sequences end with difficult task SST or CoLA, whereas all bottom tasks ends in TREC. This finding is similar to the findings from curriculum learning (Bengio et al., 2009): training a hard task later in a sequence has overall less error, and leads to better generalisation. Forgetting of a task is inversely proportional to its difficulty level, resulting in CoLA and SST having the least forgetting when added to the end of the sequence.

Task Sequence	$\min(F_{Seq}^{100,400,900})$
TREC_SUBJ_CoLA_SST	0.63
TREC_SUBJ_SST_CoLA	0.78
SST_TREC_SUBJ_CoLA	0.81
CoLA_SUBJ_SST_TREC	1.3
SST_CoLA_SUBJ_TREC	1.4
CoLA_SST_SUBJ_TREC	1.4

Table 6: The top three (green) and bottom three (red) task sequences with F_{Seq} for Layer = 1. For each task sequence, minimum F_{Seq} was considered across dimensionalities $100, 400, 900$. Most of the top task sequences finish with SST or CoLA, and bottom ones with TREC.

11 Discussion and Limitations

In our study, we used a very loose definition of what makes a task difficult, mainly comparing single-task performance on a simple CNN/LSTM model with SOTA. Our current analysis shows that task and task sequencing plays a pivotal role in forgetting observed in the network. To establish what quantifies a difficult task in a continual learning setup would require extensive experiments with a large number of varied tasks.

For task sequence TREC_SUBJ_CoLA_SST, CoLA's performance improves after training on SST, resulting in negative F_{CoLA} and a slight drop in F_{Seq}. This task sequence and network with $hdim = 900$ is the only instance where we observed $F_i < 0$. Our two-task experiments saw forgetting of 0.16 when training SST after CoLA. This observation could be a consequence of training on TREC or SUBJ beforehand. It would be interesting to gain more insights on what enabled improvement on CoLA while training SST in this task sequence.

Our results with contextual ELMo embeddings are intriguing, as the amount of forgetting is vastly

Second Task → First Task ↓	CoLA	SST	SUBJ	TREC
CoLA	—	0.16	0.38	0.39
SST	0.36	—	0.40	0.57
SUBJ	0.35	0.17	—	0.46
TREC	0.09	0.08	0.15	—

Table 7: Forgetting on sequentially training two tasks; Layer= 1, Dimensionality = 100.

Second Task → First Task ↓	CoLA	SST	SUBJ	TREC
CoLA	—	10.6	11	16.4
SST	16.2	—	11.8	16.2
SUBJ	17.8	10.6	—	14.6
TREC	15.2	11.0	11.8	—

Table 8: Number of epochs used for training the **second task** in the sequence; Layer = 1, Dimensionality = 100.

different when ELMo's parameters are fixed, versus when they are fine-tuned for each task. Our experiments favour using fixed embeddings. This finding also points at the importance of developing new specialised fine-tuning approaches similar to the one introduced in ULMFit (Howard and Ruder, 2018). When fine-tuning ELMo embeddings, CNNs have less forgetting than LSTMs, and contrastingly LSTMs have less forgetting when ELMo embeddings are fixed.

12 Conclusion

We carried out an empirical study on catastrophic forgetting, observing that LSTMs forget more than CNNs. Further experimentation provided the insight that max-pooling helps CNNs alleviate abrupt forgetting. Our findings with pre-trained embeddings suggest one should avoid fine-tuning pre-trained embeddings in a continual learning setup. We also observed that more capacity doesn't help in reducing catastrophic forgetting, and that training a difficult task towards the end of a task sequence is beneficial.

Acknowledgement

We want to thank the anonymous reviewers for their insightful suggestions, which gave rise to the experiments with pre-trained embeddings.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Robert M French. 1991. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference*, pages 173–178.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Z. Li and D. Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947.

David Lopez-Paz et al. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7765–7773.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489.

Cuong V. Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. 2019. Toward understanding catastrophic forgetting in continual learning. *CoRR*, abs/1908.01091.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019.*, pages 7–14.

Roger Ratcliff. 1990. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4555–4564.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. 2013. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318.

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

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

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomás Kociský, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.