Curriculum Based Continual Learning on Seq2Seq Language Models.

| Ramki Ramamurthy |
| --- |
| *Luddy School of Informatics, Computing, and Engineering,* |
| *Indiana University,* |
| *Bloomington, IN 47408, USA.* |
| ramrama@iu.edu. |

***Abstract* - People have the ability to constantly learn new things. Outdated and infrequently accessed data can be replaced with new data, whereas critical and frequently used data is protected. The emphasis in lifelong learning has primarily been on accumulating knowledge across various tasks and addressing the challenge of catastrophic forgetting in the realm of artificial learning systems. To overcome the limitation of catastrophic forgetting, various mechanisms have been proposed and one of the methods is to selectively slow down the learning on important weights and when training data is passed in an easy to difficult curriculum fashion similar to human learning procedure this helps the language models to better capture the statistical significance. In this research, When this curriculum learning approach combined with continual learning of tasks, we tried to study the improvement in the performance of the language model on the downstream tasks.**

I. Introduction

Transfer Learning in deep learning models has become a predominant technique. It allows the reusing of the pretrained model by fine tuning the model to focus on a particular problem. The weights of the network that has learned task A is transferred to learn a new task B and this helps the model to employ the patterns learned in solving the related task instead of starting the learning procedure from scratch. However, this limits the scope of the model to improve its performance on a specific task or domain and reduces its overall previous knowledge acquired.

Large language models have emerged as transformative tools in recent years, revolutionizing natural language understanding, generation, and a wide range of applications in a variety of domains. As the demand for these sophisticated language models grows, so does the demand for comprehensive resources to support their development, training, and fine-tuning, resulting in an increase in the planet's carbon footprint. Everytime, for fine tuning purposes the model has to be retrained and the size of the large pretrained models increased from millions to billions to improve learning on different tasks. However, Humans and animals appear to be able to learn continuously, as opposed to neural networks. The goal is to aggregate gained knowledge from various tasks, which is typically accomplished through model weight sharing, leading to the development of a single model with limited memory that excels at executing all previously learned tasks.

We chose the Transformer framework because of its strong performance across a wide range of tasks, including machine translation, document generation , and syntactic parsing. We embrace the Transformer architecture for its adept handling of long-term dependencies in text via structured memory, inspired by the success of models such as the GPT-2 decoder and T5 encoder-decoder. Unlike alternatives like recurrent networks, this architecture provides a more organized memory, improving transfer performance across diverse tasks.

We are inspired by traversal-style approaches for language modeling and subsequent fine-tuning on downstream tasks such as summarization, translation, and paraphrasing. Task-specific input adaptations are used, with structured text input treated as a unified sequence of tokens. Our experiments demonstrate the efficacy of these adaptations., allowing us to fine-tune the model with minimal adjustments to the pretrained architecture.

(Bengio et al., 2009) pioneered Curriculum Learning (CL) in the field of machine learning. The identification of simple examples is predetermined in this approach, and a curriculum ranging from simple to difficult is systematically organized to guide the learning process. According to recent research, an effective Curriculum Learning (CL) strategy can steer the learning process towards a more favorable local minimum in the parameter space by removing the negative influence of challenging or potentially noisy examples during the initial phases of training. In this approach, We tried to adopt the elastic weight consolidation similar to the synaptic consolidation and trained the model with curriculum based learning of data starting from easy ones and gradually moving on to difficult ones would benefit the model in learning process and improve its understanding better.

II. Previous works

Bengio et al. initially proposed Curriculum Learning in 2009, illustrating its advantages through toy experiments in image classification and language modeling. They advocated for conceptualizing curriculum as a collection of training criteria, underscoring the importance of uniformly reweighting examples with the target distribution at the curriculum's conclusion, shaping the development of the Curriculum Arrangement algorithm.(Guo et al., 2018; Wang et al., 2019; Platanios et al., 2019; Tay et al., 2019) employ a two-step process in curriculum formation: initially assessing the difficulty and subsequently sampling examples into batches accordingly. Notably, the evaluation methods for difficulty vary significantly across different target tasks. (Benfeng Xu 2020) demonstrated the effectiveness of CL in the context of finetuning LM’s on NLU Tasks. They introduced a Curriculum Learning (CL) framework comprising a Difficulty Review method and a Curriculum Arrangement algorithm. This framework eliminates the need for human pre-design and demonstrates high generalizability across a wide range of tasks.

There were three types of approaches that addressed the problem of catastrophic forgetting. First, by introducing changes to the architectural changes to the network. (Rusu et al., 2016; Lee et al., 2016) simply copied the entire network for the previous task and added new features for each task. The architectural complexity gets compounded with this method as the number of tasks increases. Additionally, functional approaches advocate for analogous predictions between the previous and new tasks. In (Li & Hoiem 2016) study, a technique resembling knowledge distillation is employed to foster similarity between the predictions of the previous task's network and the current network when applied to data from the new task. With data-centric approaches, information from the new task is leveraged to estimate the performance achieved in previous tasks. This method proves effective when there is minimal mismatch in data distribution across tasks.Ultimately, within model-based approaches, our attention is directed towards the network parameters rather than relying on task-specific data. Each parameter of the model undergoes the computation of an importance weight, and when training on a new task, a regularizer is introduced to penalize changes to these importance weights.

In the Elastic Weight Consolidation study (J.Kirkpatrick 2017) , this process is executed through an estimation of the diagonal of the Fisher information matrix. Meanwhile, in the Synaptic Intelligence research (Zenke 2017), importance weights are dynamically computed during training in an online fashion. In the Memory Aware Synapses MAS (Aljundi 2018) approach, they suggest examining the sensitivity of the learned function as opposed to focusing on the loss. This choice streamlines the setup significantly because, unlike the loss, the learned function is not confined to a local minimum, thereby circumventing issues related to gradients approaching zero. Moreover, These approaches have not been implemented within the domain of natural language processing (NLP) for language models to evaluate their performances on the downstream tasks like syntactic parsing, summarization, language modeling, paraphrasing and translation.

III. Method

In our method, I have used two distinct sets of experiments to investigate the capabilities of a GPT-2 model, with a focus on its decoder component. We began by putting the model through a language modeling task with the Tiny Stories dataset, which was generated through curriculum learning. This dataset was created specifically for use with a curriculum learning strategy, and an additional Elastic Weight Consolidation (EWC) loss was applied during training to improve the model's adaptability and performance. Following that, the model was trained on a summarization task using the CNN-DailyMail dataset, focusing on the first 10,000 data instances. This stage, like the language modeling phase, involves an additional round of training with an EWC loss to refine the model. Finally, we evaluated the model's performance in both language modeling and summarization tasks.

The EWC loss was included to mitigate catastrophic forgetting, allowing the model to retain knowledge gained during the initial phases of training. We sought to improve the model's overall flexibility and robustness by systematically exposing it to diverse tasks and employing curriculum learning strategies, with a particular emphasis on evaluating its proficiency in language modeling and summarization. These experiments were designed to elicit information about the model's adaptability and performance across a variety of natural language processing tasks, shedding light on its potential for transfer learning.

In the second experiment, I have used the memory aware synapses technique and instead of using the Fisher information matrix diagonal, I’ve opted for an alternative method and used the T5 encoder-decoder transformer, and the focused on implementing the Memory Aware Synapses (MAS) sequence learning task loss in fine-tuning tasks. Specifically, the T5 encoder-decoder architecture is employed for three sequential fine-tuning tasks: Translation, Summarization, and Paraphrasing. The objective is to apply MAS optimization to enable the model to learn these tasks sequentially without forgetting the information acquired from preceding tasks. This approach emphasizes the retention of task-specific knowledge, contributing to the model's adaptability and robust performance across diverse natural language processing tasks.

1. *Datasets:*

*TinyStories Dataset:* TinyStories is an artificial dataset of short stories designed exclusively with words understandable by typical 3 to 4-year-olds, generated by GPT-3.5 and GPT-4. This unique resource allows the training and evaluation of language models (LMs) with fewer parameters (below 10 million) or simpler architectures, featuring only one transformer block. Despite their simplicity, these LMs exhibit fluency, coherence, and robust reasoning skills, producing diverse and well-structured paragraphs with near-perfect grammar. TinyStories illustrates the potential for achieving linguistic proficiency and reasoning abilities in LMs without the necessity for large model size or complexity. The dataset serves as an excellent tool for investigating the emergence and dependence of language model capabilities, such as summarization and commonsense reasoning, on factors like model architecture and data distribution.

*CNN-Daily Mail Dataset:* CNN/Daily Mail, comprises human-generated abstractive summary bullets sourced from CNN and Daily Mail news stories. These summaries are structured as questions, concealing one entity within them, and the corresponding news story passages serve as the anticipated responses to the questions. The dataset, encompassing 286,817 training pairs, 13,368 validation pairs, and 11,487 test pairs, was generated utilizing scripts provided by the authors. Within the training set, source documents average 766 words over 29.74 sentences, while the summaries have an average length of 53 words and 3.72 sentences. Recognized as a valuable asset for training and assessing text summarization models, this dataset empowers researchers to explore and experiment with advanced techniques in this specific domain.

*WMT-16 En-De Dataset:* The WMT-16 En-De (English-German) dataset is a resource commonly used in the field of machine translation and language processing research. Released as part of the Workshop on Machine Translation (WMT) in 2016, this dataset focuses on the translation task from English to German. It consists of parallel corpora containing sentence pairs in both English and German, with the English sentences serving as the source and the corresponding German sentences as the target.

*Para-NMT 5m processed Dataset:* This dataset contains over 5 million English-English sentential pairs. The pairs are generated automatically by translating the on-English side of a large parallel corpus using neural machine translation.The paraphrases are generated through a back-translation process, where the original sentences from the Czeng1.6 corpus.this dataset offers a substantial repository of semantic knowledge to enhance subsequent tasks in natural language understanding.

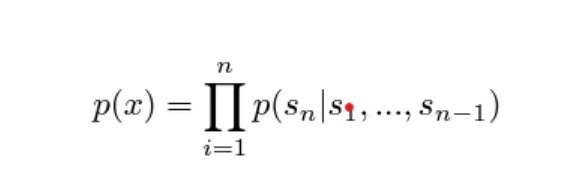
1. *Preprocessing steps*

GPT-2 tokenizer utilizes the Byte-Pair Encoding (BPE) subword segmentation. This tokenizer is designed to convert raw strings into integer sequences. Notably, the GPT-2 tokenizer, grounded in byte-level BPE, has been trained to handle spaces as integral parts of tokens, resembling the behavior of sentencepiece. Consequently, encoding of words varies based on their position within a sentence, considering the presence or absence of spaces. BPE, as a compression technique, replaces common pairs of consecutive bytes with a unique byte, offering a balanced approach between character and word-level encodings. This characteristic proves advantageous in managing vast corpora vocabularies. By leveraging GPT-2 tokenization, I aim to tokenize the TinyStories dataset and assess its performance in a language modeling task. The gpt2 tokenizer is initialized from the pretrained gpt2 tokenizer in huggingface repository. I have used Gpt2 Tokenizer to process the first 211980 samples from the tinyStories Dataset and preprocessed the text into input\_ids and attention mask. These data are evaluated based on the average perplexity metric and arranged in a curriculum ordered fashion of easy to difficult examples.

For the tokenization of datasets encompassing CNN-DailyMail, Para-NMT-5m processed, and the WMT-16 English-to-German dataset, I took the first 10000 examples as train and 1500 examples as test and employed the T5 tokenizer which comprises 32,128 subword tokens. Leveraging the SentencePiece tokenizer, T5's vocabulary extends beyond the specified 32,128 subword tokens, demonstrating its capacity to generate a larger and more nuanced vocabulary. SentencePiece, employed in the T5 tokenization process, is a versatile subword tokenizer and detokenizer designed for Neural Network-based text processing systems. Renowned for its simplicity, efficiency, and language independence, SentencePiece offers features such as lossless tokenization, customizable character normalization, self-contained models, and on-the-fly processing capabilities. Utilizing T5 tokenization, I conducted an evaluation on the downstream tasks across these datasets, harnessing the advanced subword tokenization capabilities inherent in the T5 default vocabulary and the robust processing capabilities offered by SentencePiece.

*C. Implementation Details*

**GPT 2 Architecture:**

Language modeling task is posed as the unsupervised distribution estimation from a set of examples(x1,x2,…,xn) each example consisting of variable length sequences of the symbols(s1,s2,…,sn). We factorize the joint probabilities over the symbols as a result of conditional probabilities because language has a sequential ordering.. This approach allows a controllable sampling from and estimation of p(x) as well as other conditionals of the form p(s\_n-k,….,s\_n | s\_1,…., s\_n-k-1). the learning of performing a single task can be formulated as estimating a conditional distribution of the form p(output|input). But even with the same input, a generic system ought to be capable of carrying out numerous distinct jobs. As a result, it should depend not only on the input but also on the task that needs to be completed, p(output | input, task). This method has been formulated in multitask and meta-learning settings.

Eqn-1. Language Modeling Objective.

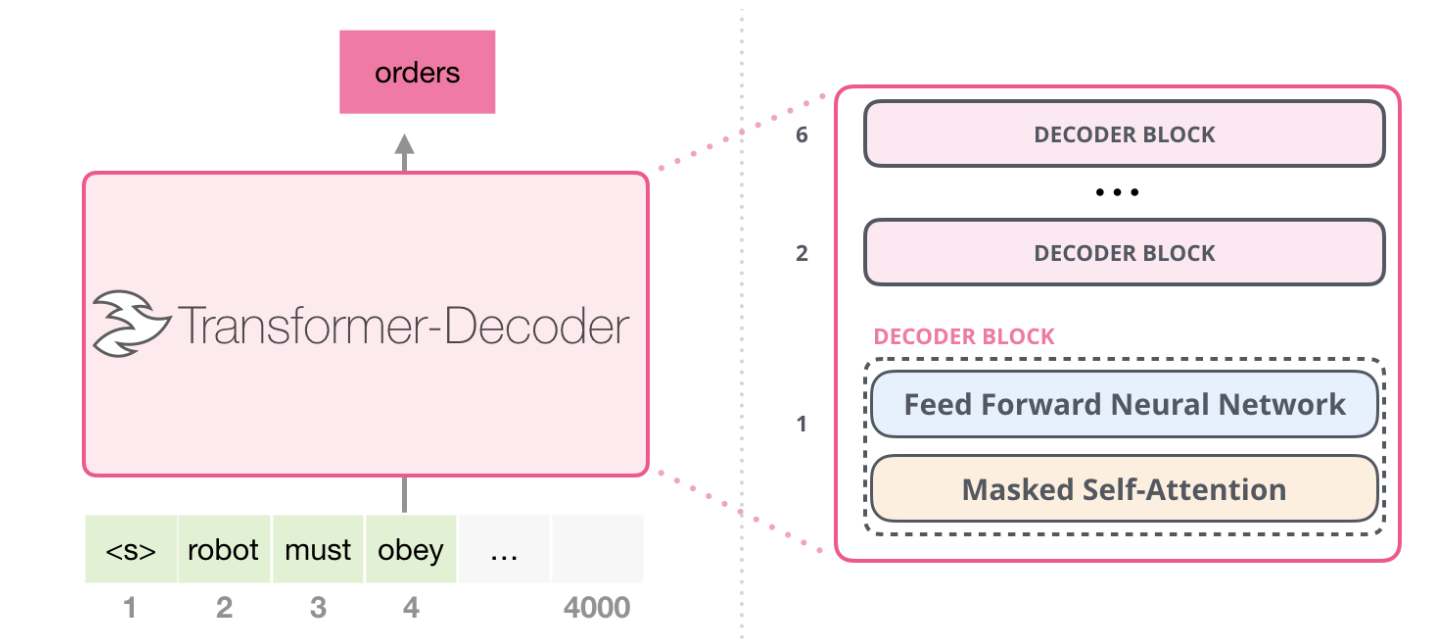
The model follows the details of the OpenAI GPT model with fewer modifications. Layer Normalization was moved to the input of each block similar to a pre-activation residual network. An additional layer normalization was added after the final self attention block. The weights of the residual layers at initialization are scaled by a factor of 1/sqrt(N) where N is the number of residual layers. The vocabulary is expanded to 50,257 and the context size can be expanded from 512 to 1024 tokens and a larger batch of 512 is used for pretraining.GPT2 uses Byte Pair Encoding to create the tokens in its vocabulary. This means the tokens are usually parts of words.

Fig 1.1 Gpt 2 Architecture.

GPT-2 employs masked self-attention rather than the self-attention employed by transformers. A standard self-attention block enables a position to peek at tokens to its right. Masked self-attention prevents this from happening, so they can only predict the next word using the left context.After the positional embedding and token embeddings are combined and fed into the model a multi-headed self-attention operation is performed on the input context tokens, followed by position-wise feed forward layers across the decoder blocks, to generate an output distribution across target tokens at the final block.

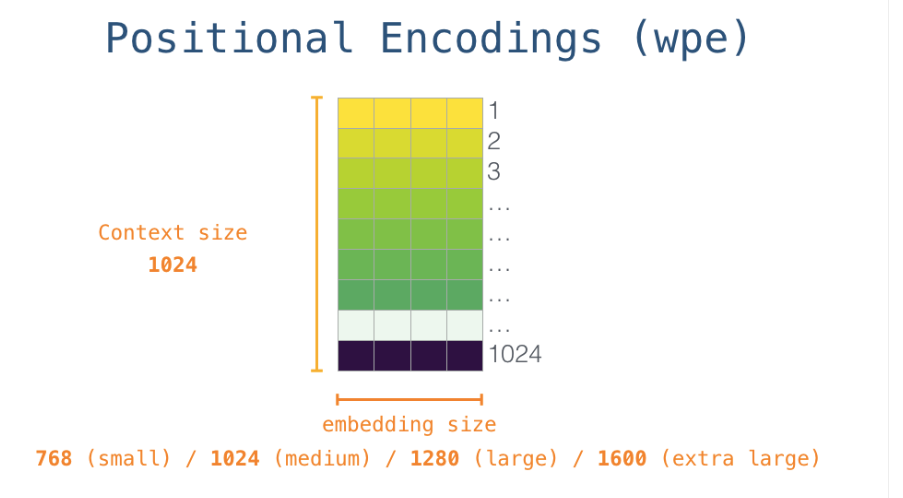
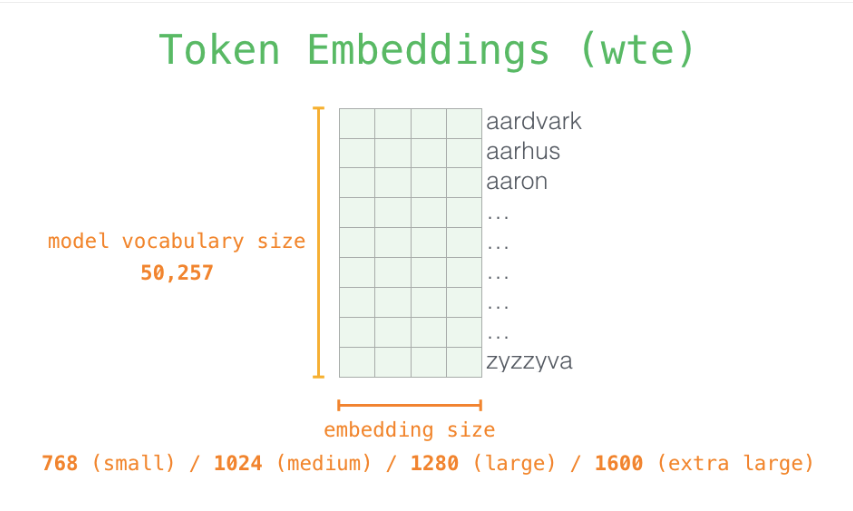
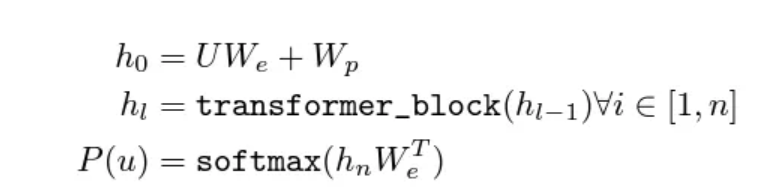


Fig 1.2 Positional Embedding and Word Embedding Sizes used in Gpt 2.

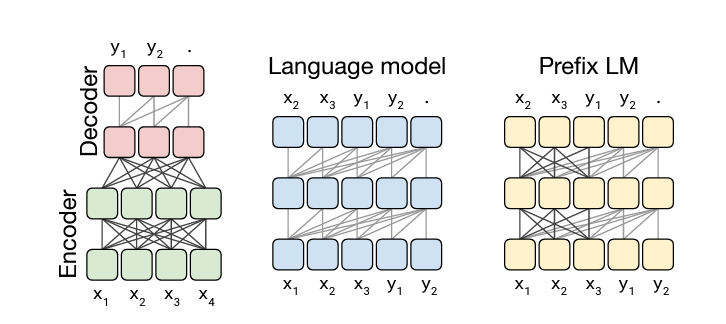
Where We is the token embedding matrix, and Wp is the position embedding matrix. The gpt2 models come in different variants as small, medium, large and extra large. For simplicity of our experiments we took gpt2-small with 768 as token embedding size and 12 decoder blocks. At the end, a language modeling head is attached with 50257 as its output features. For the language modeling task a negative log likelihood is applied for the softmax output with respect to the labels.

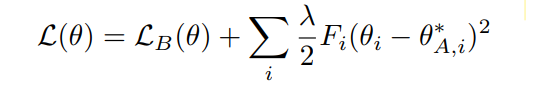
**Elastic Weight Consolidation:**

In brains, synaptic consolidation enables continual learning by reducing the plasticity of synapses that are vital to previously learned tasks. Similarly, in deep neural networks we can perform a similar operation by constraining the important model parameters to stay closer to the old values. Mastering a task involves fine-tuning the set of weights and biases (denoted as θ) in the linear projections to enhance overall performance. Many configurations of this θ will result in the same performance. This over-parameterization makes it likely that there is solution for task B θ\*\_B, that is close to the previously found solution for task A, θ\*\_A. This constraint can be implemented as a quadratic penalty.

To justify the choice of constraint and to define which weights are important the neural network training is seen from a probabilistic perspective. We can find the posterior probability of parameters given data p(θ|D) from the prior probability of the parameters p(θ) and the probability of the data p(D|θ) by using Bayes’ rule: 

If we assume the data is split into two independent parts one defining task A and other defining task B then the above equation can be rearranged as:

Thus, all the information about task A is therefore absorbed by the posterior distribution p(θ|DA). This posterior probability must contain information about which parameters were important to task A and is therefore the key to implementing EWC. but approximating the posterior of a gaussian distribution is intractable, hence instead we approximate the posterior with the mean given by parameters θ\*\_A and a diagonal precision given by the diagonal of the fisher information matrix F. thus, the Loss function for EWC becomes:



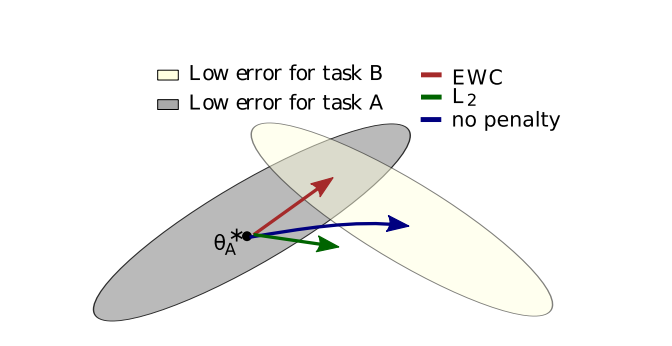
where L\_B(θ) is the loss for task B only, λ sets how important the old task is compared to the new one and i labels each parameter

Fig 1.3 Elastic Weight Consolidation ensures task A is remembered whilst training on task B.

**T5 Transformer Architecture**:

T5 stands for "Text-to-Text Transfer Transformer". The model structure adheres to a conventional and basic vanilla encoder-decoder transformer. T5 is a multi-task model designed for standard text-to-text conversion tasks. This model is pre-trained on a massive C4(common crawl web extracted text) dataset consisting of 750GB filtered raw data from various sources.In the pre-training objective of the original text, some words are dropped out with a unique sentinel token. Words are dropped out independently uniformly at random. The model is trained to predict basically sentinel tokens to delineate the dropped out text.

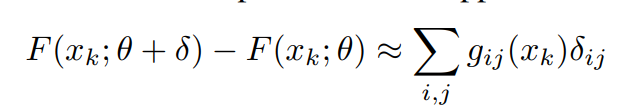
The model is trained with 3 different corruption strategies: 1.Only masking tokens without swapping them, 2. masking tokens, and substituting them with a singular sentinel token, 3. eliminating tokens are considered. The results indicate that the strategy of 'replacing corrupted spans' yields the best performance. Additionally, varying corruption rates are tested, revealing that unless a substantial corruption rate is employed (such as 50%), this setup is not sensitive to performance changes, as the model performs similarly with corruption rates of 10%, 15%, and 25%. Thus from this experiment it is observed word corruption objectives tend to work best.

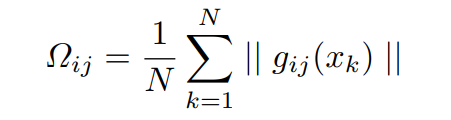
T5 is trained with a multi-task learning methodology, where the idea is to club multiple tasks while pre-training the model. These multiple tasks are further clubbed into two groups based on how they are trained, Unsupervised Training on C4 corpus and Supervised training on several NLP based tasks like question-answering, summarization, classification, etc. The model comes with different size variants: they are 1. Small, 2. Base 3. Large 4. 3B and 5. 11B. Due to the limited memory constraints, for our experimental settings, we chose the small variant with 6 layers and 60M parameters.

Fig 1.4 Left: In a conventional encoder-decoder architecture, full visibility masking is applied in both the encoder and the encoder-decoder attention, with causal masking specifically in the decoder. Middle: A language model consists of a single Transformer layer stack and is fed the concatenation of the input and target, utilizing a causal mask throughout. Right: Introducing a prefix to a language model corresponds to enabling fully-visible masking over the input.

**Memory Aware Synapses**:

In this approach, a unified neural network is employed across a series of tasks. The neural network's parameters, denoted as {θij}, represent the weights associated with connections between pairs of neurons ni and nj in successive layers. Within this model-based strategy, the primary goal is to calculate the importance weight value Ωij for each parameter θij, signifying its significance concerning the preceding tasks. In a learning sequence the model receives a sequence of tasks {Tn} to be learned each with its Training data {Xn,Yn}.For every task, there is a task-specific loss term, denoted as Ln, which will be augmented with an additional loss component to prevent the occurrence of forgetting. Upon convergence of the training process to a local minimum, the model acquires an approximation, F, of the true function, F¯. This approximation, F, is capable of mapping a novel input, X, to the corresponding outputs Y1, ..., Yn for the tasks T1...Tn that have been learned thus far.

Upon reaching convergence, the model acquires an approximation, denoted as F, of the true function F¯. This approximation, F, establishes a mapping from the input X1 to the output Y1. Preserving this mapping, F, while accommodating the learning of additional tasks is our objective. To achieve this, we assess the sensitivity of the function F's output to variations in the network parameters.By Introducing a small perturbation δ = {δij} in the parameters θ = {θij} results in a discernible change in the function output, which can be approximated by evaluating as follows:

The expression gij(xk)= ∂(F(xk;θ)) / ∂θij denotes the gradient of the learned function F concerning the parameter θij, evaluated at the data point x\_k, while δij represents the change in the parameter θij. The overarching goal is to uphold the network's predictions at each observed data point, ensuring the preservation of parameters essential for accurate predictions and guarding against detrimental modifications. We can assess the significance of a parameter by examining the magnitude of the gradient gij, indicating the extent to which a slight alteration to that parameter influences the output of the learned function for a specific data point x\_k. Subsequently, we aggregate the gradients across the provided data points to derive the importance weight Ωij for the parameter θij.

Parameters with negligible importance weights have minimal impact on the output and can be modified to minimize the loss for subsequent tasks. Conversely, parameters with substantial weights are preferably retained unchanged.

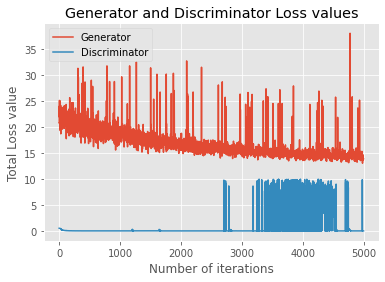
**HyperParameters**:

For the DeepSpeech2 Model we kept the learning rate as 0.00005 and the number of epochs to be 10. Also we kept the batch size as 7. We kept the number of bidirectional rnn layers to be 3 and the number of cnn layers to be 5. The rnn hidden dimension is kept to be 512. We kept a dropout rate of 0.1 in between rnn layers for better results.

The audio data is resampled to 16kHz which is the required sampling rate for the wav2vec 2.0 model. We took batch size as 5 for inference on the pretrained model and 3 for fine tuning the model. Also we kept the learning rate as 0.00005 for fine tuning and we used the AdamW algorithm to get better results along with Gradient Scalar to improve the underflowing gradients in larger models. Also we used a linear scheduler initially for a larger learning step and then gradually reduced the learning rate as the model reached optimal state.

IV. Results

After training the Patch-GAN for around 500 epochs over 5000 iterations, the figure below shows the generator and discriminator total loss values.



As we see, the discriminator loss is very low during the first 2000 iterations, then once the generator hits a loss less than 15, the discriminator loss starts to rise,and then fluctuates till 5000 epochs. This is better than the Cycle GAN-VC results, and incorporating the three new loss functions allows us to attain good performance for every speaker pair, in comparison to the baseline models.

Here is the output of fine-tuning the base Wave2Vec2 pre-trained on 960 hours of speech on the clean and noisy dr-vctk train dataset.

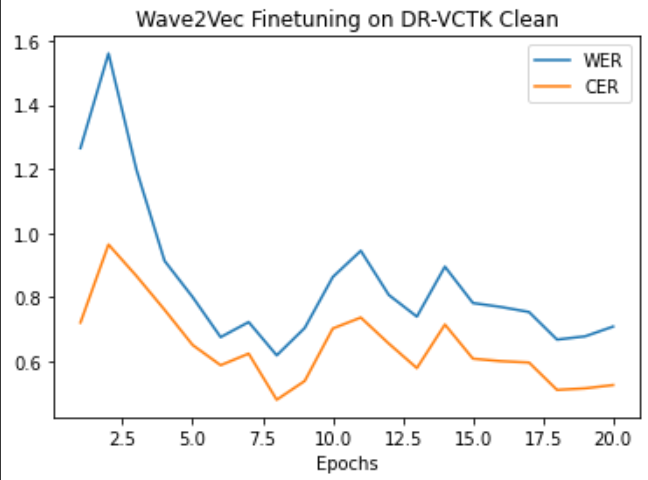


Fig 5 dr-vctk clean train metrics during fine-tuning

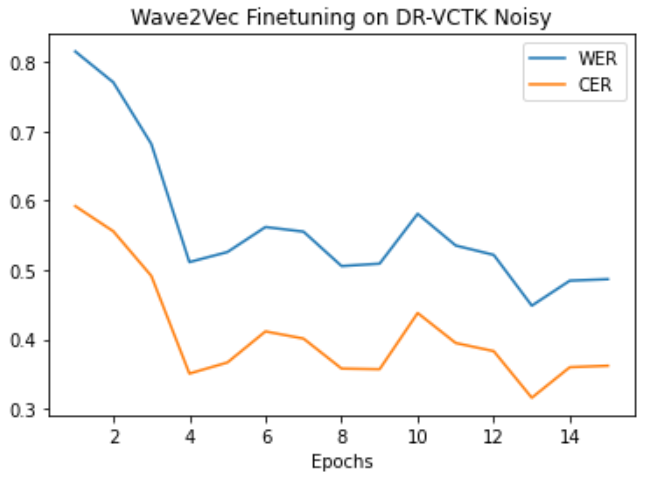


Fig 6 dr-vctk noisy train metrics during fine-tuning

Another training output we would like to show is the loss curve of our deepspeech 2 model during training on the clean and noisy dr-vctk train dataset.

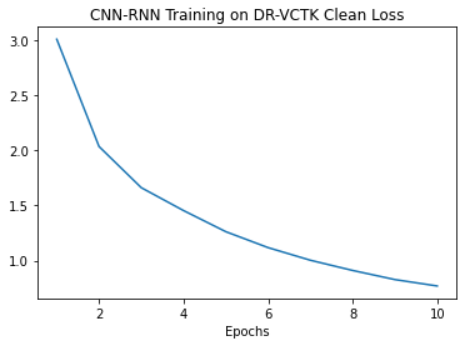


Fig 7 Deepspeech 2 CTC loss on clean dr-vctk data

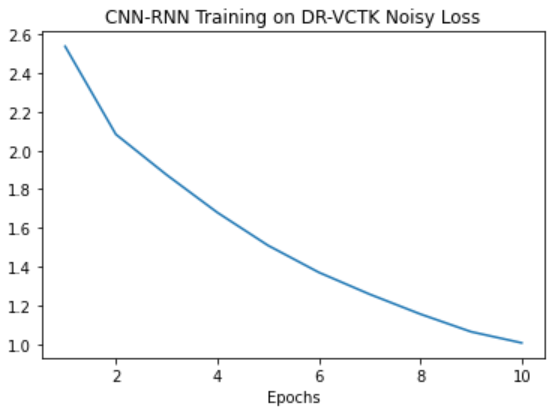


Fig 8 Deepspeech 2 CTC loss on noisy dr-vctk data

After training our models, we ran the dr-vctk test data both clean and noisy through these three models for evaluation.

|  | | dr-vctk test dataset | |
| --- | --- | --- | --- |
| clean | noisy |
| wave2vec-  pretrained  (greedy) | wer | .38 | .82 |
| cer | .03 | .18 |
| wave2vec-  pretrained  (beam) | wer | .32 | .75 |
| cer | .03 | .19 |
| wave2vec-  fine-tuned  on train  (greedy) | wer | .04 | .10 |
| cer | .01 | .05 |
| deepspeech  (greedy) | wer | .66 | .81 |
| cer | .28 | .44 |

Fig 9 ASR models performance on test dr-vctk data

Here the best performing model was the fine-tuned Wave2Vec2 as expected. The worst performing model was our deep speech 2 model. The main cause of this was the lack of training data, training hours, and the lack of language model or lexicon to give context and transition probabilities. The noisy data was harder for the ASR models to transcribe compared to the clean data. CER was always lower than WER, showing that some word errors were simple character flips due to the lack of language understanding.

Once we had tested our ASR pipeline, we wanted to evaluate the performance of our Cycle-GAN model on accented speech. This was done by feeding in accented Indian and English speech from vctk.

|  | | indian | | english | |
| --- | --- | --- | --- | --- | --- |
| raw audio | cycle-gan | raw audio | cycle-gan |
| wave2vec-  pretrained  (greedy) | wer | .48 | .94 | .46 | .94 |
| cer | .03 | .23 | .04 | .24 |
| wave2vec-  pretrained  (beam) | wer | .40 | .85 | .42 | .84 |
| cer | .02 | .22 | .03 | .23 |

Fig 10 Cycle-GAN models performance on ASR task

Contrary to our hypothesis that the GAN would transfer an accent to the indian speech and give a clearer input to the ASR pipeline yielding a lower error rate, the error rates are much higher for both the generated indian and english speech. Listening to the audio gives us the impression that instead of transferring an accent our GAN model has transferred the style of voice to the other speaker's sample, and in the process introduced a lot of noise. The error rate of the raw accented speech is also similar in the beginning. It hasn’t been shown that voice or accent transfer help ASR error rates. The noise we added mostly harmed our performance. It has been shown that noise is very harmful.

V. Conclusion and Future work

Our approach was to convert heavy accented speech to clearer accented speech by using a Cycle-GAN to do style transfer on audio data from the VCTK dataset and transcribing the speech using state of the art models like Wave2Vec2 and DeepSpeech2. Our best ASR model was the fine-tuned Wave2Vec model with a 4% error rate. Our DeepSpeech model struggled, but showed improving performance as it trained. As future work, if we can train this model on a larger dataset for a much longer time, the performance will greatly improve. We also could implement a beam search solution, with a language model to improve the error rate further.

The Cycle-GAN performed poorly when connected to the pre-trained Wave2Vec model and evaluated. It seems to have added quite a bit of noise when listening to the output. From seeing the nearly equivalent word error rate of the Indian and English speech, it could be an issue that our dataset didn’t have heavy accented speech that would cause the ASR model issues, or it could be that changing accents isn’t as important as other factors. The Cycle-GAN did succeed in some aspects, such as transferring speaker A’s voice to speaker B’s sample and vice-versa. Future work might include finding a different dataset that has more challenging accents.

Our initial idea was to use Cycle-GAN or some existing GAN architecture to perform accent transfer between two input audio. Unfortunately, we were unable to get proper output and our results were very noisy, and we worked on performing a voice transfer with minimum Mel-cepstral distortion (MCD) and modulation spectra distance (MSD) values. We could also work on improving our Cycle-GAN by trying to do accent transfer instead of voice transfer, and removing the noise generated. We could try to use the noisy and clean speech in our dataset as input to our Cycle-GAN to investigate using GAN’s for denoising speech, as noise clearly as shown to be a heavy detriment to ASR models.

VI. Contributions

We all worked on the project together and contributed equally. The literature review, dataset pre-processing, model implementations, designing test and acquiring results, and writing the presentation and final report were done collaboratively.

VII. References

|  |
| --- |

|  |
| --- |