IDS 576 - Deep Learning and Applications
Final Project Report - Hateful Memes

Rohith Kokkula Aarjav Sanghvi Aditya Chaudhari Medhavi Pokhriyal

Abstract

Through this report, we have tried to implement a deep learning model for binary classification for the 'hateful memes dataset' by Facebook Research. The hateful memes dataset is constructed such that a single unimodal architecture (only text or only image) struggle to classify and only where multimodal models (combination of text and image architecture) succeed. We managed to implement the model through two ways- first by the baseline code provided by DrivenData (the website which was hosting the hateful memes challenge) where a fast text and resnet architecture was trained separately and then combined; second, by pretrained model provided by MMF (Facebook Research Team) where the training was done on the hateful memes dataset.

1 Introduction

The popularity of Internet memes grew as a result of current improvements in social media platforms and content-sharing websites, and today's Internet environment includes websites devoted just to meme sharing. Memes started off as largely comical, but the concept evolved through time, and now they span a wide range of topics, including political and social commentary. Because of the widespread use of memes and their ability to transmit condensed themes, they have become a popular tool for disseminating hate speech directed at specific individuals or groups.

Some of the most popular memes, in example, incorporate both visuals and text, making them difficult for automated systems to comprehend. Because hate speech is intrinsically multimodal, any autonomous hate speech detection system would need to take both text and visual information into account in order to accurately describe and forecast the presence of hateful speech. On the other hand, both of these sorts of information must be processed together, because the text or visual content may not reflect hateful content on their own, but when combined, their message changes.

The hostile portion of the meme is usually linked via subtle language or external knowledge of the real-world occurrence. In that respect, the malicious meme challenge is more of a multimodal categorization problem involving visual-linguistic information. Furthermore, hostile memes cover a wide range of topics and frequently include unusual items that no off-the-shelf classifier would recognize, which differs significantly from the ordinary objects seen in conventional multimodal datasets. Injury, traditional costume use by the subject, and historical settings are difficult to recognize and have a small sample size. Even without the lingual aspect, these variables make the visual understanding portion of the issue difficult to solve.

2 Hateful Memes Dataset

The Hateful Memes dataset is a so-called challenge set, with the goal of fine tuning and testing large-scale multimodal models that have been pre-trained, for example, by self-supervised learning. The dataset's primary principle is "quality over quantity."

Facebook AI created the dataset to encourage well-developed multimodal modeling methods. The dataset contains 8500 photos, each of which has a unique id and a label that indicates whether the image is toxic or not.

Benign Confounders

A benign confounder is a minimum replacement image or replacement text that flips the label for a given multimodal meme from hateful to non-hateful.

For example, in the "love the way you smell today" example, the image of the skunk might be replaced with a rose image, or the text could be replaced with "skunks have a very distinct smell." In both circumstances, the meme would no longer be derogatory, but rather a statement, and the label would be reversed. These innocuous confounders make the dataset more hard, and any system that wants to handle this challenge must be multimodal: relying solely on text or images would result in poor results.

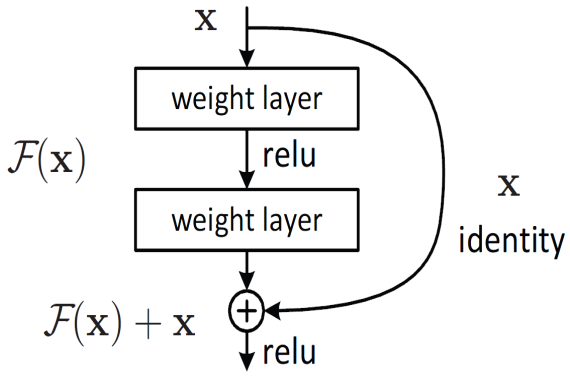
3 Literature Review

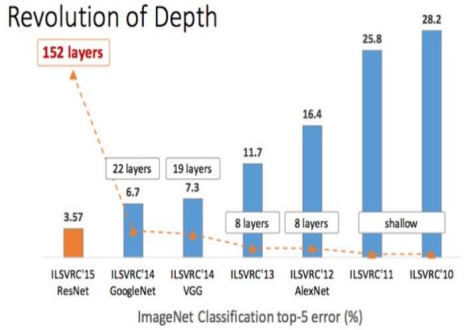
3.1 Unimodal

When we are working with a single data type to create a model, that type of model is called unimodality. For instance, text completion system - our input data for training the model is text and consequently, the output received from the model is the text as well implying we are working with a single data type throughout the process.

a. **ResNet**

While solving a complex neural network problem we add some additional layers to the network to improve accuracy and performance. This addition of additional layers helps learn more complex features. However, it has been observed that there is a maximum threshold for depth in the normal CNN model and the increasing number of layers on top has degraded the performance of the model. ResNet is a deep learning neural networking model which uses residual blocks and solves this problem for very deep neural networks. Skips connections are a major part of these residual blocks which skip several layers in the network and therefore solves the vanishing gradient problem. Moreover, these skip connections also aids the model in learning the identity functions which makes sure that the higher layers would be at least as good as the lower layers, if not better. This prevents model degradation as residual blocks make it easy for the layers to learn identity functions.





In plain networks the output is given by; H(x)=f(x).

Therefore, in order to learn an identity function, f(x) should the same as x, where the equation for ResNet would be;
H(x) = f(x) +x
f(x) = 0
H(x) = x (we get x as the output which is also our input)

The performance of neural networks with additional layers has been greatly improved by using ResNet, as shown in the plot of error percent when compared to neural networks with simple layers.

b. **FastText**

FastText is a small toolkit that can be used to create scalable text representation and categorization solutions. It runs on normal, generic hardware and, thanks to a feature that decreases the amount of memory used by fastText versions, it can even fit on cellphones and low-end machines.

It includes models that have been pre-trained using Wikipedia and is available in 157 different languages. fastText can be used as a command line, a library, or a link to a C++ application for use cases ranging from prototyping to production.

c. **BERT**

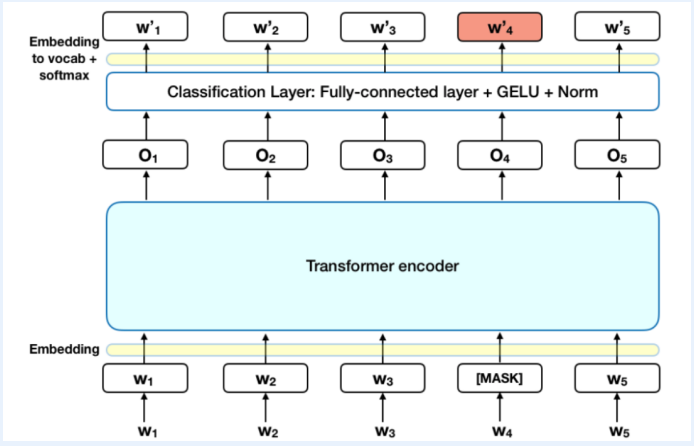
(BERT) is an acronym for Bidirectional Encoder Representations from Transformers. BERT is a program that uses surrounding text to help computers grasp the meaning of ambiguous words in text. BERT is built on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are determined dynamically based on their connection.
The Transformer encoder reads the complete sequence of words at once, unlike directional models that read the text input sequentially (left-to-right or right-to-left). As a result, it is classified as bidirectional, however it is more correct to describe it as non-directional. This property enables the model to deduce the context of a word from its surroundings (left and right of the word).

BERT is pre-trained on two independent but related NLP tasks using this bidirectional capability: Masked Language Modeling and Next Sentence Prediction. The goal of Masked Language Model (MLM) training is to hide a word in a sentence and then have the software guess which word was hidden (masked) based on the context of the hidden word. The goal of Next Sentence Prediction training is to have the software predict whether two provided sentences have a logical, sequential connection or are just random.

- **MLM:**

  15 percent of the words in each word sequence are substituted with a [MASK] token before being fed into BERT. Based on the context provided by the other, non-masked words in the sequence, the model then attempts to predict the original

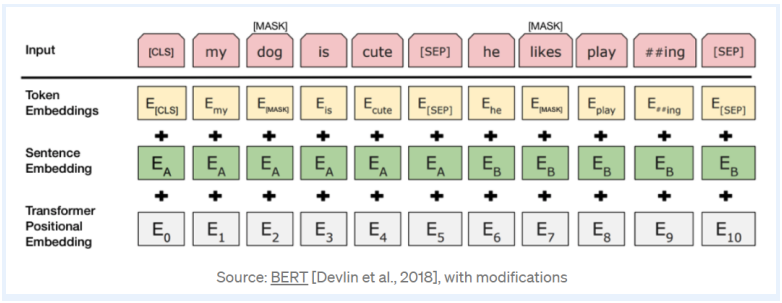
value of the masked words.



- **Next Sentence Prediction (NSP)**

The BERT training approach involves feeding the model pairs of sentences and learning to predict whether the second sentence in the pair is the next sentence in the original document. During training, 50 percent of the inputs are a pair in which the second sentence is the following sentence in the original text, while the other 50 percent are a random sentence from the corpus. The random sentence will, it is assumed, be detached from the first sentence.

1. Before entering the model, the input is processed in the following way to assist the model distinguish between the two sentences in training:
2. At the start of the first sentence, a [CLS] token is inserted, and at the end of each sentence, a [SEP] token is placed.
3. Each token has a sentence embedding that indicates whether it is Sentence A or Sentence B. Sentence embeddings are similar to token embeddings in concept, but with a vocabulary of two.
4. Each token is given a positional embedding to denote its place in the sequence. The Transformer paper explains the concept and implementation of positional embedding.



3.2 Multimodal

When we are working with multiple data types to create a single model, the process is called multimodality. For instance, image to text - our input data to the model will be an image and the expected output is in text format. For example, we feed the image of a mountain to our model and in return the model returns “mountain” in text format. Other similar examples would be voice to text conversion and text to audio conversion .

a. Visual bert :

VisualBERT is a vision and language model that is multimodal. Visual question answering, multiple choice, visual reasoning, and region-to-region correspondence activities can all be done with it. To prepare embeddings for image-text pairs, VisualBERT employs a BERT-like transformer. The text and visual characteristics are then projected onto a latent space that has the same dimensions as the visible space. Each image is run through a pre-trained object detector, and the areas and bounding boxes are extracted to feed the model. As visual embeddings, the authors use the features acquired after feeding these regions through a pre-trained CNN like ResNet. They also include absolute position embeddings and feed the generated vector sequence to a BERT model. As in BERT, the text input is concatenated ahead of the visual embeddings in the embedding layer and is assumed to be constrained by [CLS] and [SEP] tokens. The textual and visual portions' segment IDs must also be configured suitably. The text is encoded using the BertTokenizer. To obtain the visual embeddings, a bespoke detector/feature extractor must be employed. We next add a co-attention layer that interacts with each other after incorporating transformers for each of the input data, text, and image. After that, we add numerous levels of transformers and a co-attention layer; the more layers there are, the better the results. Finally, we get the similarity between the two input data and this tells us if the caption is related to the image.

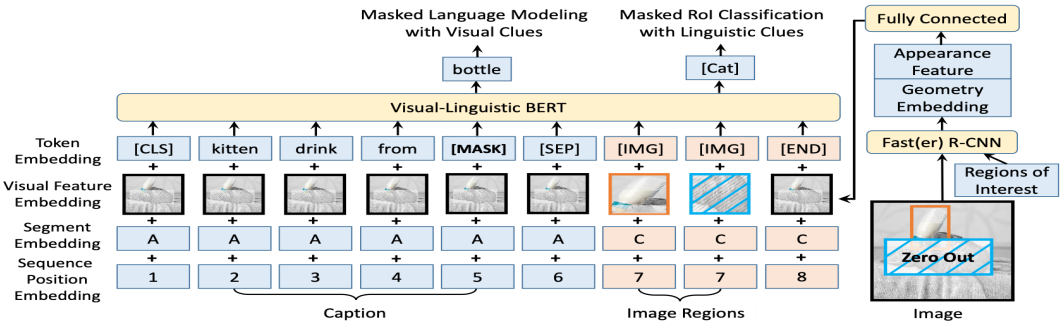


Figure 1. Architecture for Pre-training VL-BERT

4 Methodology

Some of the most popular memes, in example, incorporate both visuals and text, making them difficult for automated systems to comprehend. Because hate speech is intrinsically multimodal, any autonomous hate speech detection system would need to take both text and visual information into account in order to accurately describe and forecast the presence of hateful speech. On the other hand, both of these sorts of information must be processed together, because the text or visual content may not reflect hateful content on their own, but when combined, their message changes. Our research provides an end-to-end architecture that can assess both text and visual information derived from Internet memes automatically. We use a FastText-based architecture for text processing, and a ResNet for picture processing. Several fully connected layers concatenate and process the outputs of these layers. We can begin the model-building process now that we have a concept of how we'll need to analyze the data. As we create the model, there are three big picture issues to keep in mind.

- Dataset handling
- Model architecture
- Training logic

4.1 Baseline Model

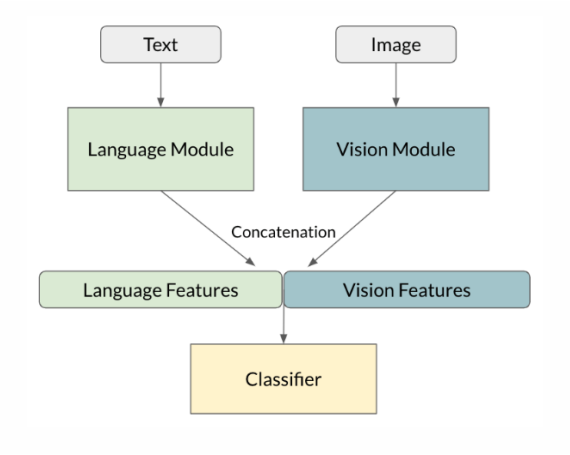
Models and applications for vision: Popular datasets, model architectures (including pretrained weights), and basic image transformations are all included in PyTorch's torchvision.

Models and utilities for languages. Facebook's fasttext AI makes it simple to learn data embeddings. It's a nice starting point before moving on to more sophisticated methods like transformers.

We will use a torchvision vision model to extract features from meme images and a fasttext model to extract features from extracted text belonging to images. These language and vision features will be fused together using torch to form a multimodal hateful memes classifier.

At first, we perform image transformation to resize the images to make it appropriate for our training model. Thereafter, we transform the images to tensors. Once the images are a uniform same size, we can make a single tensor object out of them. Once our data was ready, we moved on to build the model.

We first mapped the images to the captions using keys for this purpose. A high level view of the model creation process is as follows:



To fuse these two models together, we use a library called pytorch_lightning - This library takes a piece of code (mainly for-loops) and converts it into code snippets (wraps that code in a single line of code). All we have to do is add data and enter values for the hyperparameters.
There are three main steps that were implemented before we proceeded with training:

1. Data Processing - this process is encapsulated in the HatefulMemesDataset class.
2. Multimodal fusion - This process is encapsulated in the LanguageAndVision class.
3. Training, early stopping, checkpoint saving, and submission building - this process is encapsulated in the HatefulMemesModel class.

Then we defined values for our hyperparameters before we moved forward with training the model.
Thereafter we went on with training our model. The advantage of using the pytorch_lightning library is that it will automatically capture the best checkpoint.

4.2 MMF

MMF is a modular framework for vision & language multimodal research from Facebook AI Research (FAIR).
The three main features of MMF are:

1. Less Boilerplate - By offering boilerplate code for distributed training, common datasets, and state-of-the-art pretrained baselines out-of-the-box, MMF was built from the ground up to allow us to focus on our model.
2. Powered by PyTorch - MMF is built on top of PyTorch, giving us access to all of its capabilities. As a result, we can apply all of our PyTorch skills here.

3. Modular and Composable - MMF is designed to be expendable and reusable. We can use certain MMF components that we care about, thanks to the modular design. MMF can simply adjust to our demands thanks to the customizable system.

Using MMF's encoders, modules and utilities, we are building a fusion model which fuses ResNet pooled grid features with fasttext embedding vectors to classify a meme as hateful or not hateful.

MMF acts like a wrapper that has a pretrained model loaded in the library. All we have to do is call a subsequent function to access the model, assign values to the parameters and run the model.
Steps taken to implement the model:

1. Convert the downloaded zip data file to MMF required format.
2. Then we perform data visualization on the downloaded dataset.
3. Train a model by accessing the pretrained model in MMF (the pretrained model used here comprises ResNet and fasttext).
4. Check for prediction using the trained model.

5 Challenges

Owing to the complexity of the project there were several challenges we have faced-

1. One of the dependencies for pytorch_lightning was discontinued and the previous versions were not available on the web anymore. Hence we had to replicate the logic based on what we could find in the forums in order to create our baseline model.
2. We did not have a powerful machine to train such a computationally demanding model. Hence, we turned to google colab. However, the system RAM used to get fully occupied and then the training process would crash. To circumnavigate this issue, we reduced the batch size from 64 to 16 (we tried 32 but that didn't work either) .
3. We also tried running the above model on our local windows machine but for some reason we kept getting permission denied error. We tried several things but none of those methods seemed to work. Finally we were able to run the model on the macOS

6 Results and Inference.

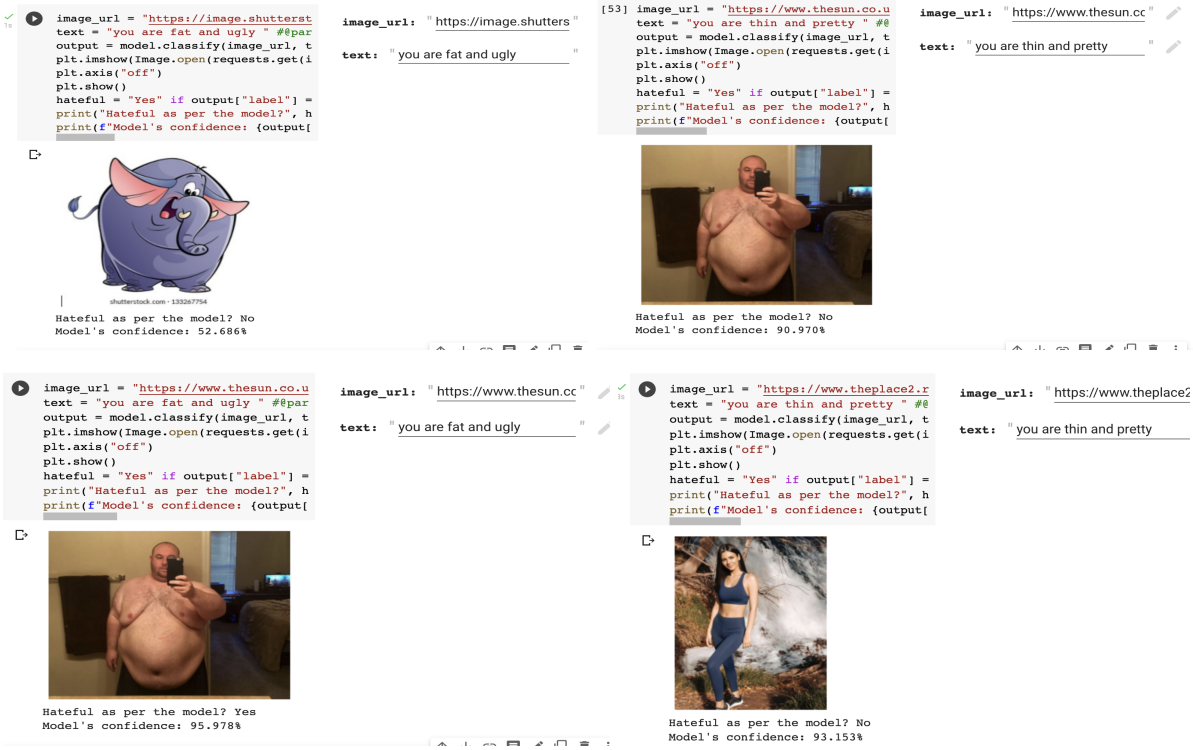
6.1 Easy Examples

Here we are trying to see how mmf's pretrained MMBT model is working with images and text. In the images below we see by changing text and image we can see how its confidence is changing.

In the first image of an animated elephant with text- 'you are fat and ugly' model confidence is 52% with not offensive -this is expected as the image has not relation to the text

In the second image of a fat man with a sarcastic comment of you are thin and pretty the model predicts it to be not hateful with a high confidence of 90%. Here we can it can be argued that the MMBT is not trained/advance enough to understand the sarcasm, but it is correct in predicting that it is not offensive

In the 3rd and 4th image of the fat man and the women with a text of 'you are fat and ugly' and 'you are thin and pretty' respectively, we see that the model does a good job in predicting that correct label with a high confidence.



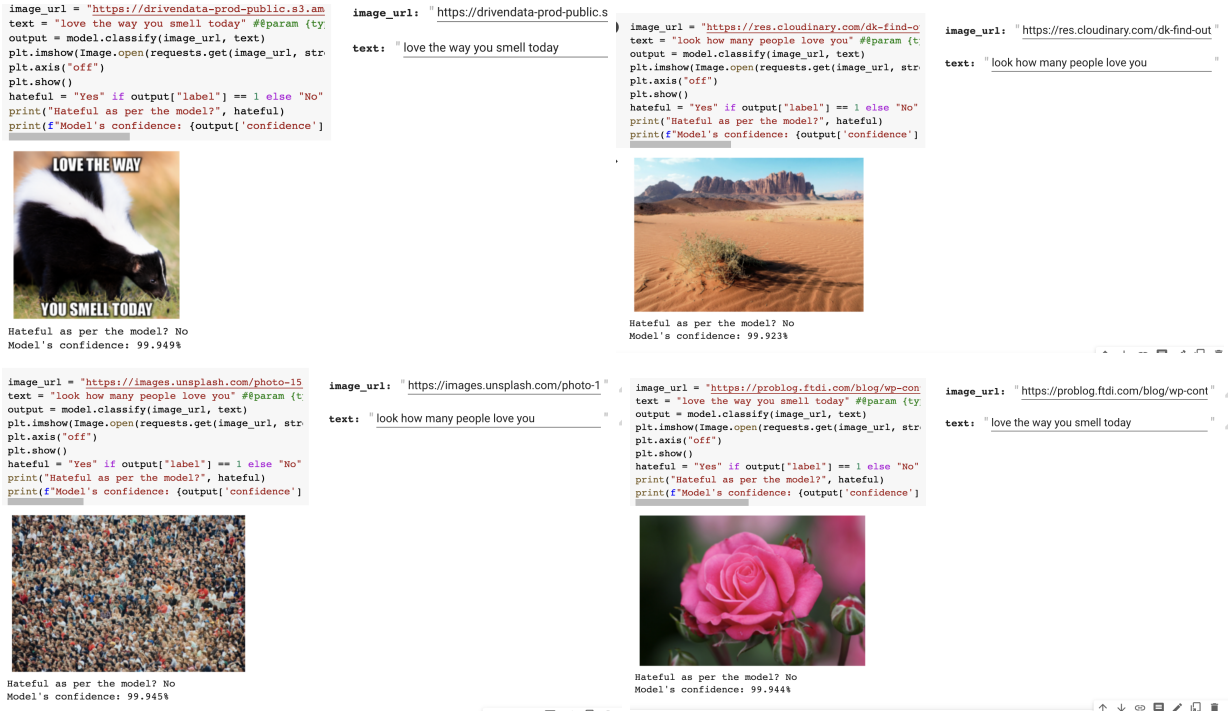
Overall, from our brief analysis through these 4 examples, the MMBT model is somewhat able to correctly classify the straightforward examples

6.2 Difficult Examples -Benign Confounders

The real challenge and the main purpose of the project was to classify memes where a combination of a text and image make a meme offensive.

For the top image of a skunk and a desert with text of ‘love the way you smell today’ and ‘look how many people love you’ respectively -both of which are offensive, the model predicts as not offensive with a high confidence

For the bottom images of a crowd and a rose with text ‘look how many people love’ and ‘love the way you smell today’ respectively- both of which are not offensive, the model predicts as not offensive with a high confidence.



For the benign confounders, we see that the MMBT model is not able to classify between hateful and not hateful memes

7 Conclusion

Through this report, we have tried to understand and implement a deep learning model that uses a combination of both image trained architecture as well as text trained architecture. Using the documentation provided by Facebook research (MMF Framework) and Driven Data (baseline model) we were successfully able to load the dataset, run the code, train the model and infer the results.

8 References

1. The Hateful Memes Challenge
2. Proceedings of the ML Research
3. BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science
4. What is BERT (Language Model) and How Does It Work? (techtarget.com)
5. VILBERT.pdf (utexas.edu)
6. Introduction to ResNet
7. FastText: Meta AI
8. How to build a multimodal deep learning model to detect hateful memes - DrivenData Labs
9. Installation | MMF
10. https://arxiv.org/pdf/2005.04790.pdf