Development of an Explainable AI System For Text Classification

Report of Summer Internship 2020

Syomantak Chaudhuri
Electrical Engineering Department, IIT Bombay
syomantak@iitb.ac.in

Under the guidance of

Prof. Gitta Kutyniok
Mathematics Department, TU Berlin
kutyniok@math.tu-berlin.de

19th July 2020

Contents

1 Introduction

With the rise of neural networks, there has been a lot interest in trying to understand the decision made by complex networks having millions of parameters. Despite good performance, treating neural networks as a black box is not very convincing in critical applications such as defence and medicine where a wrong decision might be very costly. Explainable AI (XAI) is a branch of machine learning which focuses on explaining the decisions of these complex models. In the process, we may find out various facts about the models, such as whether the model is actually focusing on what we as humans perceive to be important in the classification or whether the model is actually biased and only focusing on some meta-data present in the training images. There are several ways to interpret these complex neural networks and we have particularly focused on a rate distortion based approach.

We explored the various possible implementations of rate distortion (RDE) [1] based XAI approaches for NLP task. The loss function used in the RDE paper can be changed a little to see how it affects the performance. As the distortion measure changes, so will the stationary point of the distortion based loss function. Along with the loss function, the performance metric is itself dependent on which distortion function we choose. While mean squared distance is a possible measure, a more interpretable measure seems to be the accuracy curve based on relevant word deletion [2] (described in Section 4).

We mainly used 2 datasets to conduct the experiments : Newsgroup 20 (NG20) and Ag News. Newsgroup 20 consists of 20 classes and the documents are email exchanges between people and sometimes contain ascii art/signatures at the end of the emails. This last fact sometimes posed a challenge. Also, some emails were too short at times. The dataset itself is not very large, it contains about 11000 training documents and about 7000 test documents. AG News is a slightly larger dataset containing news belonging to 4 classes. Each class has 30000 training documents and 1900 test images. Some of the news articles seem to belong to multiple classes and there are often junk strings in between (like #36;). Additionally, there may be a bias problem as often the news source/agency is also a part of the document and the network may end up learning to recognise that agency and which class of news it may exclusively report on.

While our work did not initially focus on developing a SOTA classifier, towards the end of the internship we were toying with the idea of feedback training and knowledge distillation using XAI methods. Further, we explored knowledge distillation on image classification tasks as well.

2 Model Description

2.1 News Group - 20

We had used word2vec [3] embedding to get a 300 dimensional vector for each word. If there are a total of N words in the document, then we get a N dimensional input tensor. We then treat this just like an image with the same dimension and pass it to a convolutional network with fully connected layers at the end. We chose a 5×300 convolutional kernel with 1280 output channels. This is then followed by fully connected layers as shown in Figure 1. The model is based on [2], [4] and [5]

Note that while training, each document is made of length 400 either by concatenating 0 vectors at the end or truncating the document for ease of batching, as described by the author of [2]. However, to generate relevance maps, entire text is used without adding 0 vectors or truncating the document. This is possible due the initial convolutional and max pooling layers. Input length has no restriction.

Using this model, we achieved an accuracy of about 77%. Since we only interested in explaining the decision, the low accuracy is not a problem.

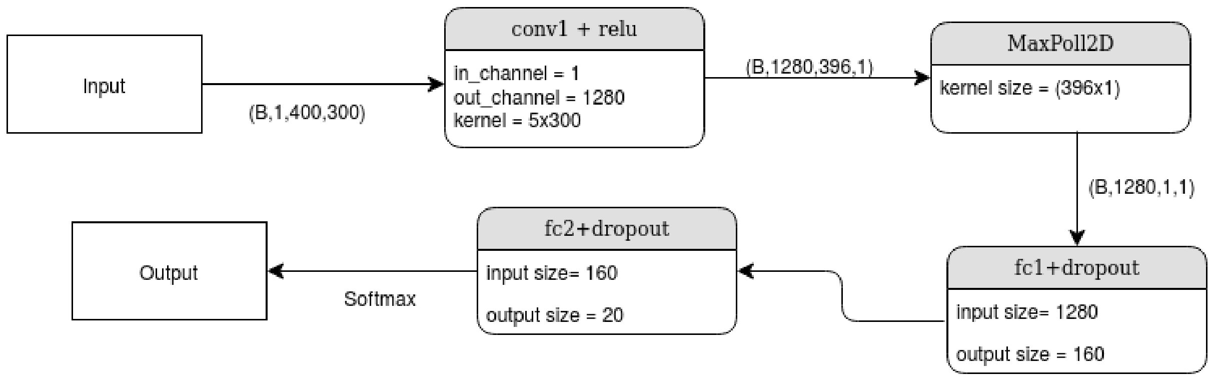


Figure 1: NG20 Model

2.2 AG News

We used fastText [6] embedding to get a 300 dimensional vector for each word. The architecture is quite similar to NG20. It is shown in Figure 2. While training, each document is made of length 80. Unlike the NG20 methods, we also make the document length as 80 during generation of relevance maps (although, this is not strictly necessary). We chose the length as 80, since the mean length of the documents was around 35.

Using this model, we achieved an accuracy of about 91%.

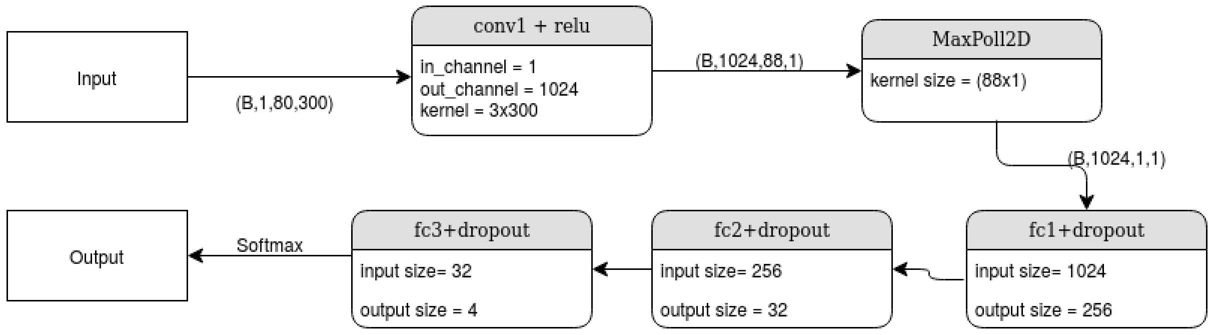


Figure 2: AGNews Model

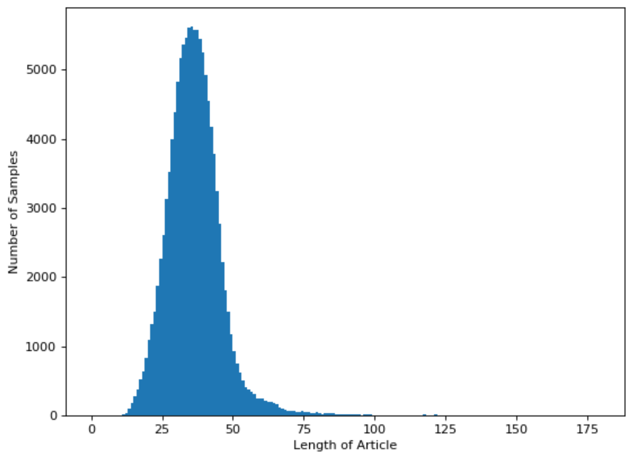


Figure 3: AG News length of documents histogram

Exact text processing should be read from the source code in Github. Ag News required some extra processing as some particular garbage strings are present in the dataset.

3 Relevance Map Generation

3.1 Method Description

We want to assign a 'relevance' value to each word in a document. The relevance value for each word is in the range $[0, 1]$. We use the NG20 model to help describe the method.
For a particular file, we denote the number of words after preprocessing as N. We denote input to the classifier as X with shape $N \times 300$. X is formed by taking the word2vec embedding of the N words.
Let $\Phi(.)$ be the classifier, i.e., we denote $\Phi(X)$ be the 20-dimensional column vector of shape 20×1. $\Phi(X)$ represents the values 20 output neurons, before softmax is applied in the model for input the X, or equivalently, we are looking at the vector which is the output of 'fc2+dropout' in Figure 1.
We aim to obtain a relevance map s of size N. Thus, a relevance value for each word is represented by the s vector. For computation, we cast s to a matrix S of shape $N \times N$ by using the diagonal matrix $diag(s)$.
Let G be a tensor of shape $N \times 300$ with each value sampled IID from the normal distribution. The values in G are clipped between $(-1, 1)$. We choose to perturb the input by replacing X with $SX + (I - S)G$, where I is the identity matrix of size $N \times N$. Therefore the output of the classifier is $\Phi(SX + (I - S)G)$.

3.2 Loss Functions

The classifier's ,$\Phi(.)$, weights are frozen in the optimization to get the 'relevance values' of each word. We define a few loss functions as follows. We discuss the intuition behind the choices at the end of the section.
The optimal value of s is obtained using mini-batch SGD. We constrain the S matrix to be diagonal during the optimization process. In a given batch of size B, B different values of G is used in each batch for gradient descent.
In some of the experiments, instead of using $G \sim \mathcal{N}(0, 1)$, we used word2vec embeddings of 64 different prepositions and articles (neutral words) of English language; this means the batch size is also fixed at 64 for such method.

3.2.1 Loss Function 2

The loss function $L_2(s)$ is defined as

$$L_2(s) = 0.5||\Phi(SX + (I - S)G) - \Phi(X)||^2 + \lambda||s||_1$$

3.2.2 Loss Function 1

$L_1(s)$ - Let $\Phi(X)_i$ refer to the i-th component of the 20-dimensional vector $\Phi(x)$ and let h be the index of the output neuron with maximum activation before softmax when the input is unperturbed, ie,

$$h = \underset{output-neurons}{\arg\max} \quad \Phi(X)$$

$$L_1(s) = 0.5(\Phi(SX + (I - S)G)_h - \Phi(X)_h)^2 + \lambda||s||_1$$

3.2.3 Loss Function 4

$L_4(s)$ - Define set M as $0, 1, 2...., 19$

$$L_4(s) = \sum_{i \neq h; i \in M} (\Phi(SX + (I - S)G)_i - \Phi(X)_i)$$
$$+\Phi(X)_h - \Phi(SX + (I - S)G)_h + \lambda||s||_1$$

3.3 Intuition Behind The Loss Functions

L_2 loss tries to ensure the output is not altered due to the changed input while L_1 loss only concentrates on preserving the output of the neuron corresponding to the true class. L_4 loss encourages deleting words which negatively affect the true prediction or increase neuron values for the other class, ie, L_4 loss is similar to other methods like LRP which give positive and negative scores. L_4 loss' relevance scores may be interpreted better by finally scaling the relevance values to $[-1, 1]$ range. (There is no L_3 loss, the weird numbering is a palimpsest of the initial days of disorganized coding!)

L_2 is the exact same function used in the original RDE paper. We found L_4 generates slightly more meaningful relevance maps visually and there is no hassle of choosing a good λ either. In L_2 loss, we also tried an adaptive learning method. We would do the optimization and get the relevance map; if the mean relevance value of the document is greater than a certain predetermined range, then the optimization is performed again with an increased λ. Similarly, we decreased λ when the mean relevance value is less than the range. This is done in a while loop till the mean relevance value falls in the desired range.

4 Performance Metrics

The performance metrics used in the RDE paper was how distortion changes with decreasing the noise and increasing the actual data. Although this makes a lot of sense in terms of rate distortion theory, the problem is that the distortion function can vary. By changing the function, we may get drastically different results. Although using L_2 loss (without the regularization term) makes sense, we tried using another performance metric we saw in the LRP paper. For the correctly classified document, we start by removing top K most relevant input words of each text and replace it with zero vector (which happens to be the word2vec embedding of 'a') or random noise; the average accuracy obtained is the y value on the graph.

Unlike LRP, the RDE based approach takes significantly more time to generate the relevance maps so instead of taking the entire test dataset to get the accuracy vs deletion graph, we took about 7 randomly correctly classified texts of each of the 20 classes in NewGroup-20 dataset.

5 Greedy Approach

Based on the accuracy metric, we can directly try to get a greedy method based approach. This becomes a combinatorial problem. Let's say we want to get the top 10 words. This is achieved by Algorithm 1.

The complexity of this algorithm can be described as $O(n^2)$ (10*n*n assuming $\Phi(X)$ is evaluated in $O(n)$ time). We can improve this by making a slight modification. Instead of looping over the entire N words, we can loop over let's say, a set of 20 'likely' words. We can get these likely words by choosing top 20 most relevant words from the relevance map by L_2 optimization. However, L_2 optimization is itself time consuming. Instead, we can try narrowing it down by choosing it from LRP maps (We haven't done this part in my experiments, we only chose the likely words from L_2 (both with and without adaptive λ),L_4 maps). Assuming LRP map is generated in $O(n)$, the greedy algorithm's complexity comes down to $O(n)$.

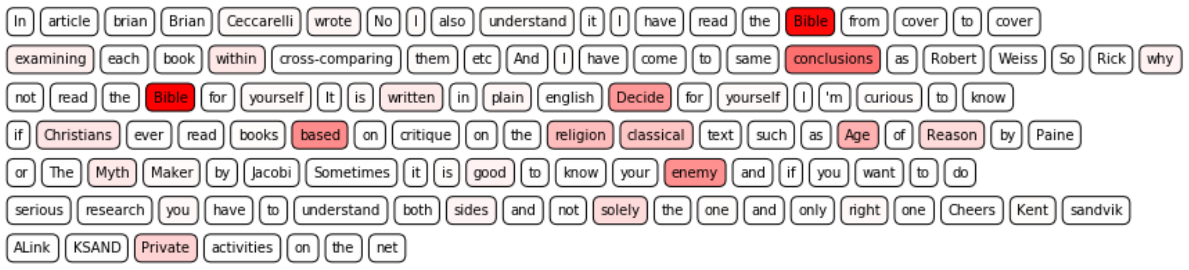
Algorithm 1: Greedy Algorithm

```
Input: X = $N \times 300$ matrix (N = number of words in the document)
Output: Rel (an ordered array)
Rel = [ ]                                              // empty array
h = argmax $\Phi(X)$                                   // h = neuron with max activation
for ($i = 0; i < 10; i++$) do
    minVal = +inf
    minInd = -1
    for ($j = 0; j < N, j \notin Rel; j++$) do
        X' = del X[j]                                  // replace j-th word with 0 vector
        val = $\Phi(X')_h$
        if $val < minVal$ then
            minVal = val
            minIndex = j
        end
    end
    X = del X[minInd]
    Rel.append(minInd)
end
```

6 Experiments

6.1 Loss 1 (NG20)

In this experiment, we used adaptive λ with L_1 loss. The distortion function was L_1 loss with the regularization term.

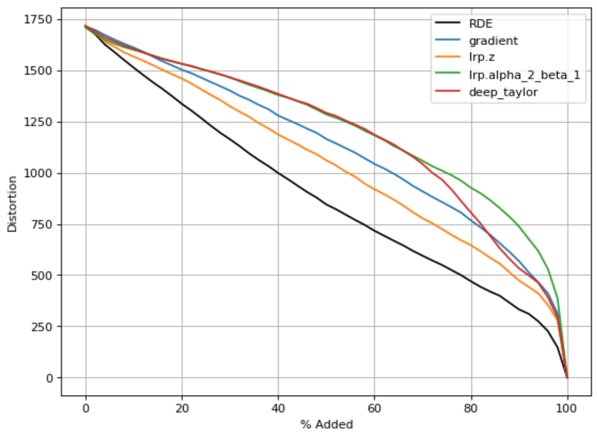


In article brian Brian Ceccarelli wrote No I also understand it I have read the Bible from cover to cover examining each book within cross-comparing them etc And I have come to same conclusions as Robert Weiss So Rick why not read the Bible for yourself It is written in plain english Decide for yourself I 'm curious to know if Christians ever read books based on critique on the religion classical text such as Age of Reason by Paine or The Myth Maker by Jacobi Sometimes it is good to know your enemy and if you want to do serious research you have to understand both sides and not solely the one and only right one Cheers Kent sandvik ALink KSAND Private activities on the net

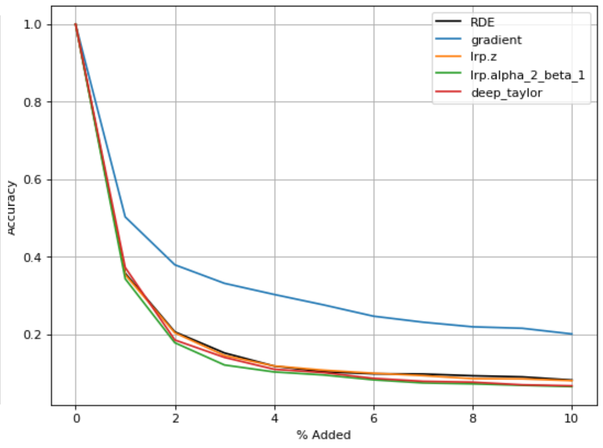
Figure 4: Heatmap - text belongs to religion.misc category

The relevance values were quite dense for this document which could have been improved by setting a better terminating range for the adaptive λ in the optimization but for the purpose of illustration, we normalized the relevance scores (divided by maximum relevance score) and then took the 8th power of each relevance, effectively, zeroing out the lesser values in Figure 4. The RDE L_1 loss based method does seem to do better in distortion curve (maybe because it is optimized on that criteria itself) but there is not much difference in terms of the batched accuracy curve. However, it is actually not competitive in the zero vector accuracy graph.

Note - In the code we have defined the L_1 loss as the loss defined in this document with the regularization term so the L_1 loss of this document is not exactly the same function as that in the code but the results shown in this document is consistent. This applies to L_2 loss as well.



(a) Distortion curve



(b) Batched Accuracy curve

Figure 5: Correction - (R) graph x axis should be labelled 'Words Removed'

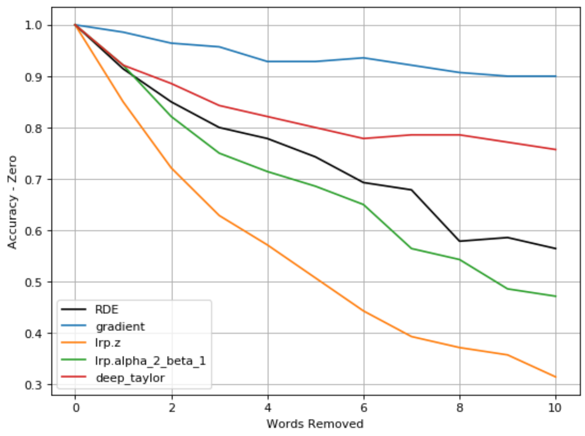


Figure 6: Zero vector replaced accuracy

Note- In accuracy graph of Figure 5b we had replaced the words with a batch of random noise as described in Section 4. We replace each word by a batch of random noise and then calculate the accuracy(call this method to get accuracy as 'batched noise accuracy'). However, in Figure 6 we used zero vector to replace the words instead of a batch of random noise(call this method to get accuracy as 'zero vector accuracy'). The zero vector graph seems to have better differentiation capability than the batched version and the authors of [2] used it as well.

6.2 Loss 2 (NG20)

This experiment, was based on adaptive λ with L_2 loss. The distortion function was L_2 loss with the regularization term.

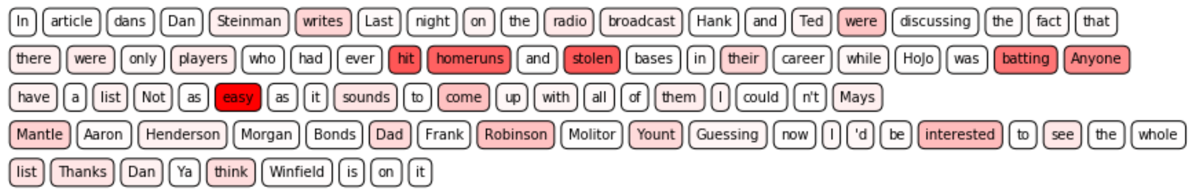
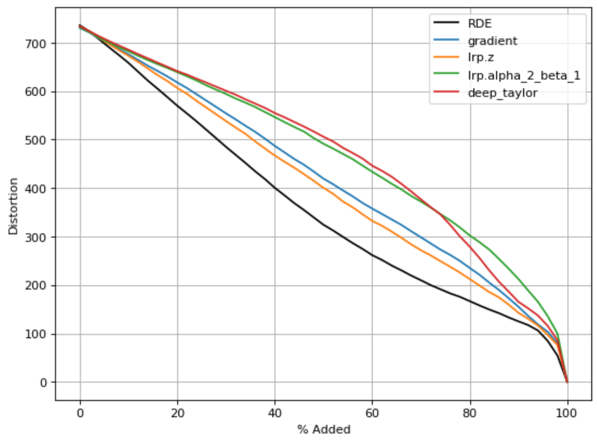
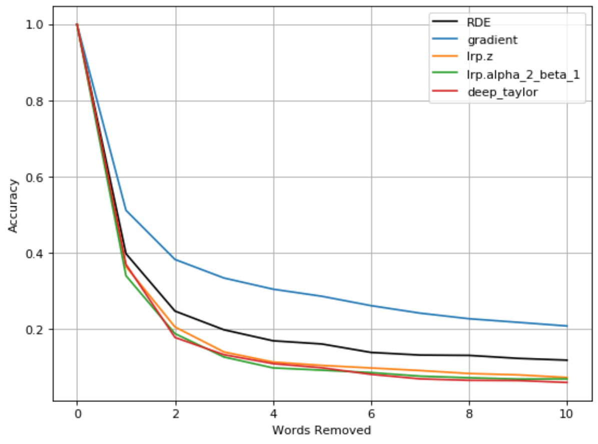


Figure 7: Heatmap - text belongs to rec.sport.baseball

Similar to the previous experiment, we normalized and took 8th power of each relevance score in Figure 7. The RDE L_2 loss based method does better on distortion curve, like the previous experiment, but is actually worse off in both 'batched noise accuracy' and 'zero vector accuracy'.



(a) Distortion curve



(b) Batched accuracy curve

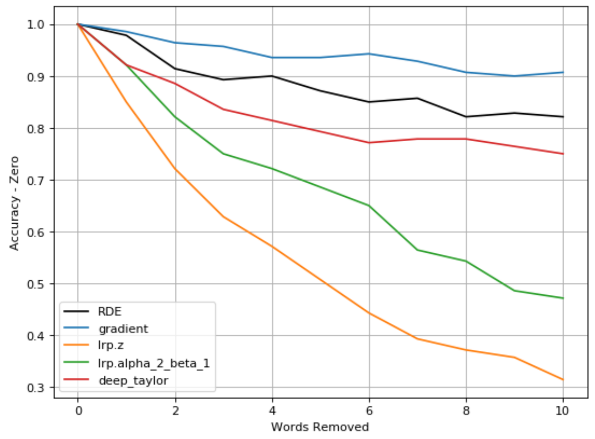


Figure 9: Zero vector replaced accuracy

6.3 Loss 4 (NG20)

We used L_4 loss in this experiment so there is no problem of setting any λ.

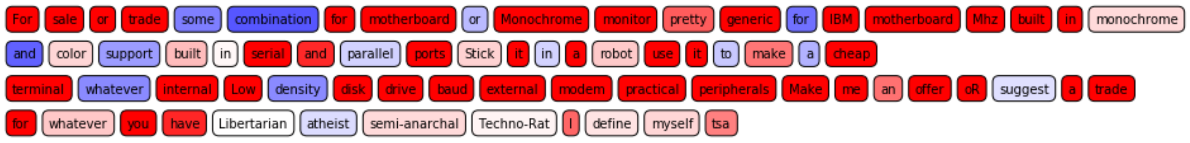
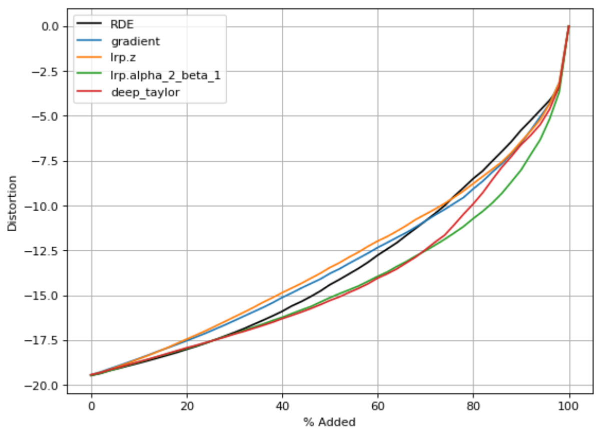
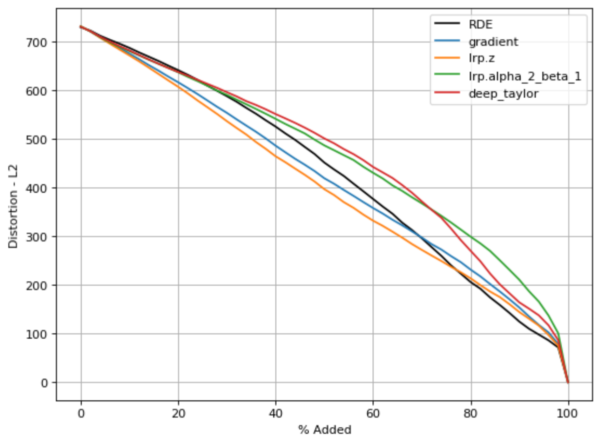


Figure 10: Heatmap - text belongs to misc.forsale

Unlike previous experiments, the relevance values here are somehow 'signed'. A low relevance value actually means that the word is negatively affecting the prediction, so the heatmap is slightly different in figure 7 (low values are now in blue, previously they were white; 'neutral' values are now 0.5 which is white). Distortion curve are made with both L_2 loss with the regularization term and L_4 loss and is shown. Since distortion based on L_4 loss function can be negative, hence, the curve does not seem to have much interpretability. The accuracy curve is also shown. The method is actually worse off compared to the other methods but it seems to offer more intuitive interpretability as explained later.
Note - Instead of using random noise vectors while optimization, we have tried using word embeddings of neutral words like prepositions in English language (call this 'batched prepostion accuracy').

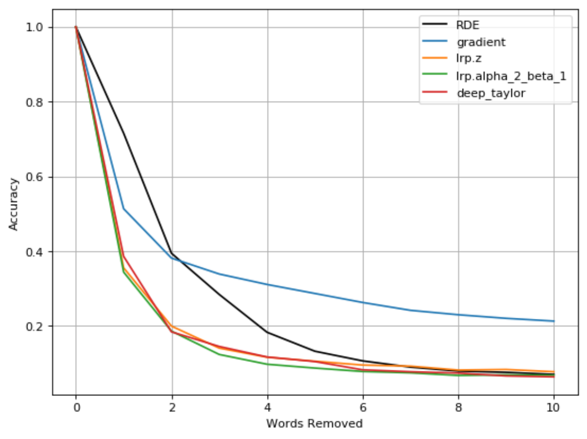


(a) Distortion curve L_4 loss

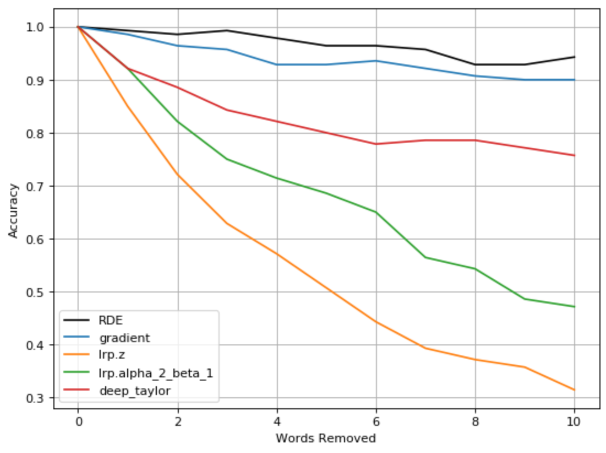


(b) Distortion curve L_2 loss

Figure 11: The 2 distortion curves



(a) Batched accuracy



(b) Zero vector accuracy

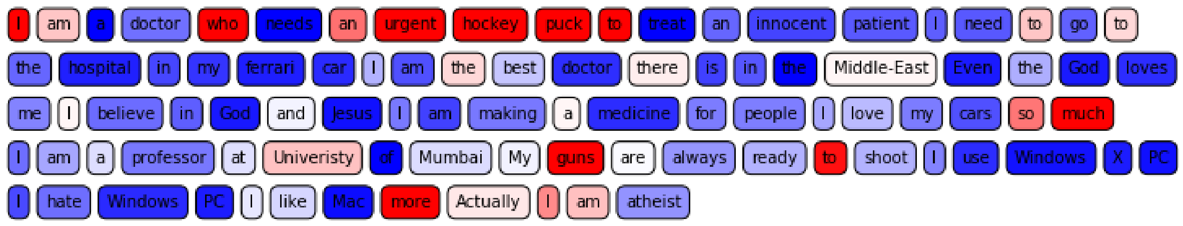

I am a doctor who needs an urgent hockey puck to treat an innocent patient I need to go to the hospital in my ferrari car I am the best doctor there is in the Middle-East Even the God loves me I believe in God and Jesus I am making a medicine for people I love my cars so much I am a professor at Univeristy of Mumbai My guns are always ready to shoot I use Windows X PC I hate Windows PC I like Mac more Actually I am atheist

Figure 13: Heatmap of a difficult text, classified as rec.sport.hockey

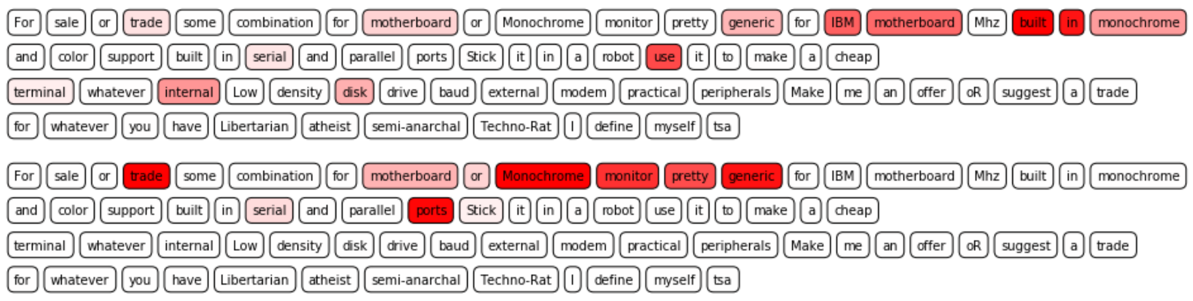
Despite not having very promising performance metrics, we find this loss function interesting as the heatmap shown in Figure 13 is much better than the heatmaps produced by LRP and deep taylor (present in the notebook on Github). Given that the network predicts the text as hockey, the method does pretty well to find hockey related words as well as negate the words which one would expect to find in other classes such as talk.politics.mideast and talk.religion.misc. Hence, this method inspires the next experiment.

6.4 Combined Loss 4+1 (NG20)

In an attempt to take the best out of L_4 and L_1 loss (good interpretability visually, good performance respectively), we tried to combine both methods. For this, top one-third of the L_4 relevant words are chosen. This yields two possible methods :

- Fix relevance of all other words as 0 and then do L_1 optimization with adaptive λ. In optimization, the L_1 norm is taken between the obfuscated output neurons and original output, ie, all relevance scores as 1. There is another possibility that we take the norm between the obfuscated output and the output when all the chosen words (top one-third) have relevance score as 1 and other words have 0; this alternate approach was not explored.
- Fix relevance of all other words as 1 and then do L_1 optimization with adaptive λ. In optimization, L_1 norm is taken between the obfuscated output neurons and original output, ie, all relevance scores as 1.

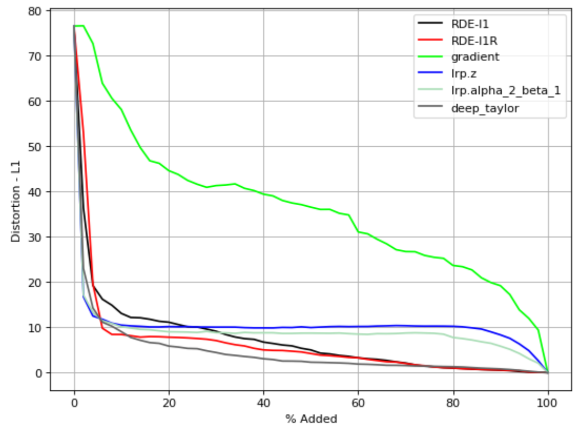
Both methods are competitive on the distortion curve but the performance is below par on both the accuracy graph, however, the heatmap (Figure 17) does show results that seem more accurate intuitively. One flaw in our line of thought might be that even though the heatmap generated below seems quite good


For sale or trade some combination for motherboard or Monochrome monitor pretty generic for IBM motherboard Mhz built in monochrome and color support built in serial and parallel ports Stick it in a robot use it to make a cheap terminal whatever internal Low density disk drive baud external modem practical peripherals Make me an offer oR suggest a trade for whatever you have Libertarian atheist semi-anarchal Techno-Rat I define myself tsa

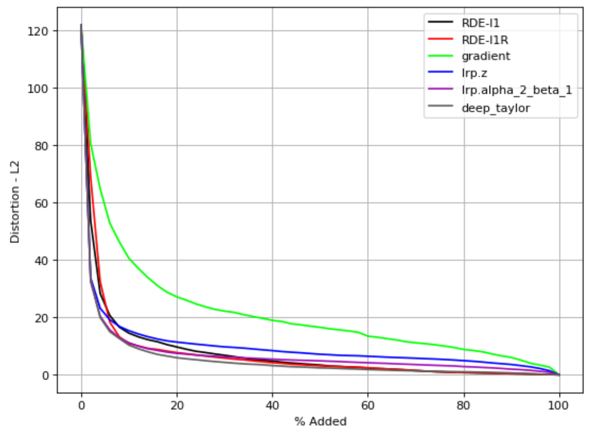
For sale or trade some combination for motherboard or Monochrome monitor pretty generic for IBM motherboard Mhz built in monochrome and color support built in serial and parallel ports Stick it in a robot use it to make a cheap terminal whatever internal Low density disk drive baud external modem practical peripherals Make me an offer oR suggest a trade for whatever you have Libertarian atheist semi-anarchal Techno-Rat I define myself tsa

Figure 14: Heatmap - text belongs to misc.forsale. Top figure is by setting all other word relevance as 1 (call this RDE-L1 method) and bottom figure is by setting all other word relevance as 0 (call this RDE-L1R method)

compared to LRP, it is a quality that is invariant of the model we choose. The heatmap represents the model's decision making and not how we perceive the document. Hence, it is possible that a particular loss function's stationary point makes more sense to us, but it does not quite correspond to being a good explanation.

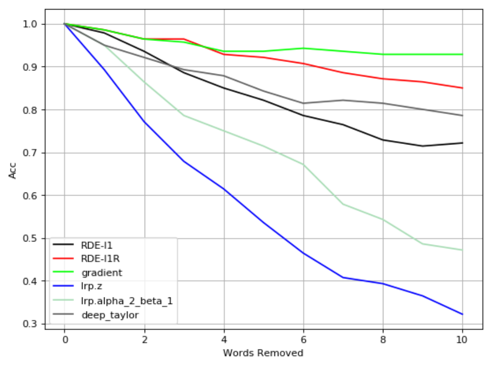


(a) Distortion curve L_1 loss

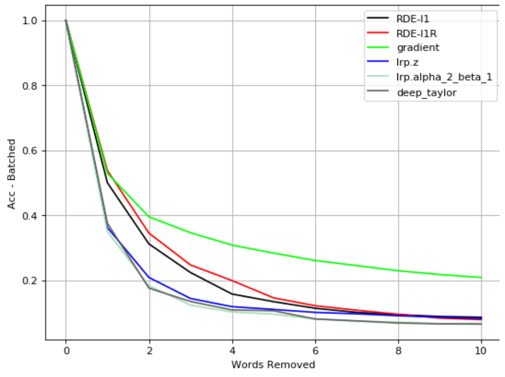


(b) Distortion curve L_2 loss

Figure 15: The 2 distortion curves



(a) Zero Vector Replacement



(b) Batched random noise accuracy

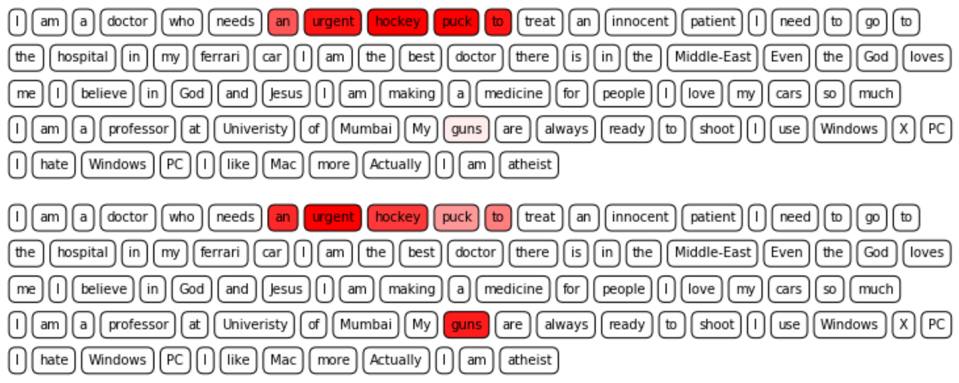


Figure 17: Heatmap of a difficult text, classified as rec.sport.hockey. Top heatmap is by RDE-L1 and bottom heatmap is by RDE-L1R

6.5 Greedy Algorithm

6.5.1 NG20

As described in Section 5, we conducted the experiment with a vanilla greedy method, greedy method combined with L_2 loss and combined with L_4 loss. The greedy method provides an ordered output, the relevance values itself are not meaningful in particular.

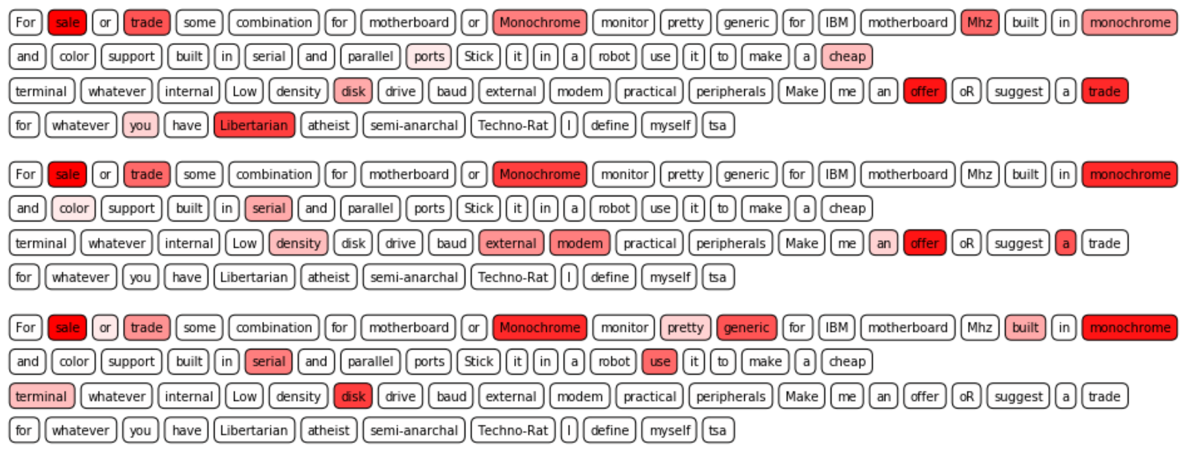
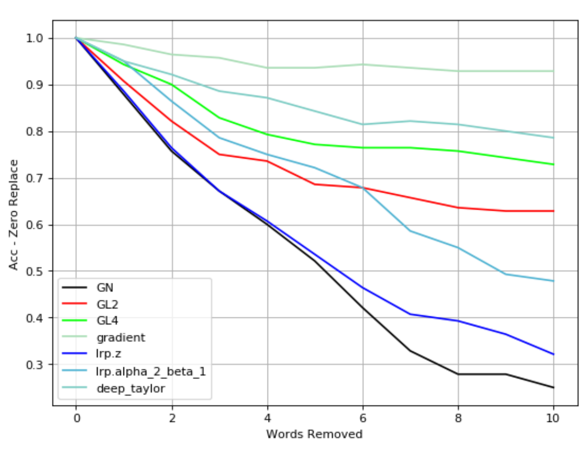
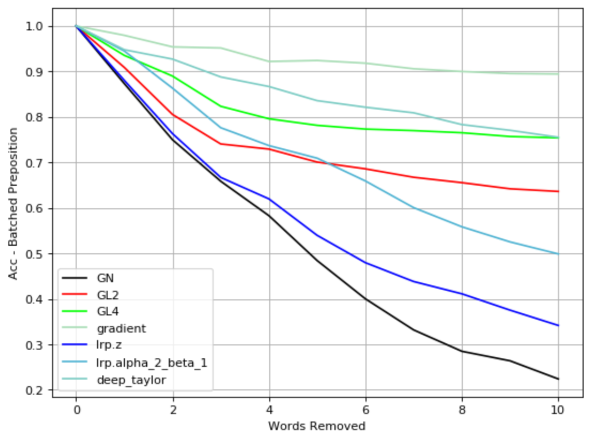


Figure 18: Heatmap - text belongs to misc.forsale. Top figure is by vanilla greedy (GN), middle figure by greedy and L_2 (GL2) and bottom figure by greedy and L_4(GL4)



(a) Accuracy with 0 vectors



(b) Accuracy batched

The accuracy curves are much better for the vanilla greedy method as compared to all the other methods so far!

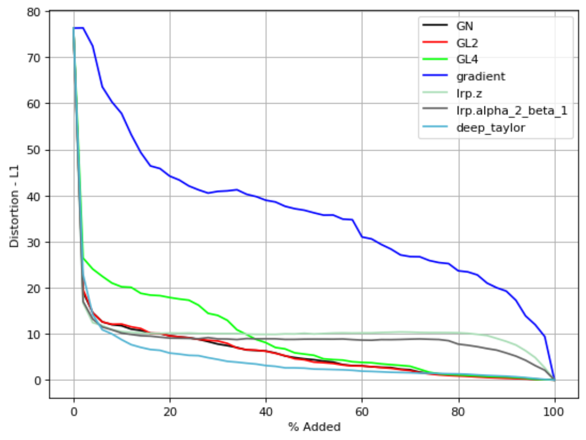


Figure 20: Distortion curve with L_1 loss

Note- In the batched accuracy graph before this experiment, we had replaced the words with a batch of random noise, but in this experiment, we used a batch of neutral words - prepositions and articles 19b. It seems like the batched version with prepositions is closer to the zero vector version than the batched version with random noise. The authors of LRP had used zero vector version, hence, we suggest that the batched-prepositions is the better batched version. In fact, since it is so close to the zero vector version of the graph, one may just use the zero vector version to save computation time.

In order to understand why the combined greedy methods don't do so well, we tried so do a few more experiment in this topic. We tried a fixed few λ for every document and we have labelled those methods as 'L2-Fixed x'. We tried adaptive λ with a few fixed intervals for each document; we have labelled those as 'L2-Adaptive x'. The label 'L4' is the same as 'GL4' previously. We have also tried some other methods as shown in Figure 21. For ease of observing, We have grouped similar methods under a common color. Based on the experiment, it seems a good interval for the adaptive λ method can lead good results although it does seem like a moot point since the vanilla greedy method is much faster.

In Figure 22, we have sampled misclassified documents from the test set and just like the normal accuracy curve, we start deleting the top relevant words corresponding to the misclassified neuron. If we

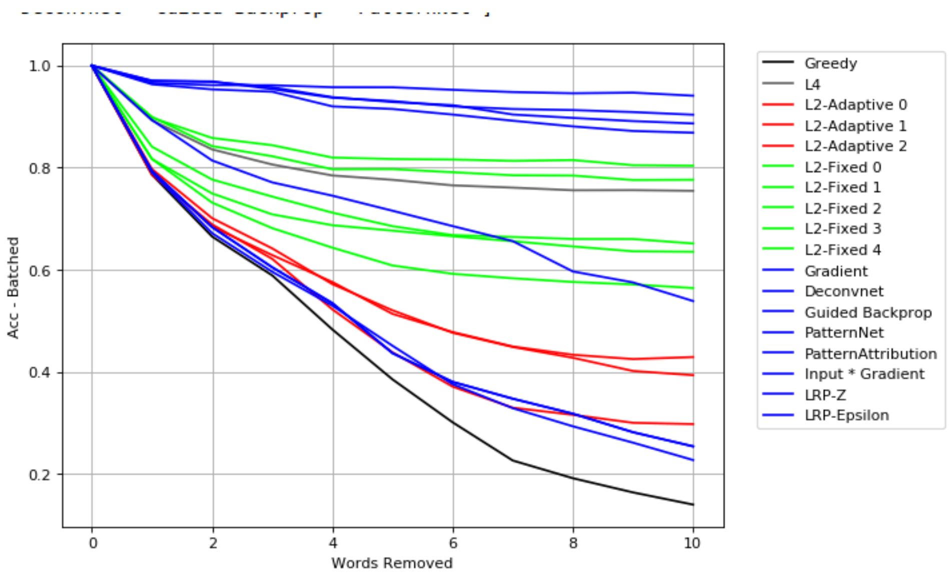


Figure 21: Preposition batched accuracy

were using just L_2 loss then this doesn't make sense as the relevance is not dependent on which neuron we choose as output, or $_h$ in the equations, but this makes sense for Greedy, L_1 and L_4 methods. In the L_2 based greedy methods, the initial set of 'likely' words chosen is indeed invariant of the output neuron but choosing output neuron has a significance in the subsequent optimization by greedy algorithm.

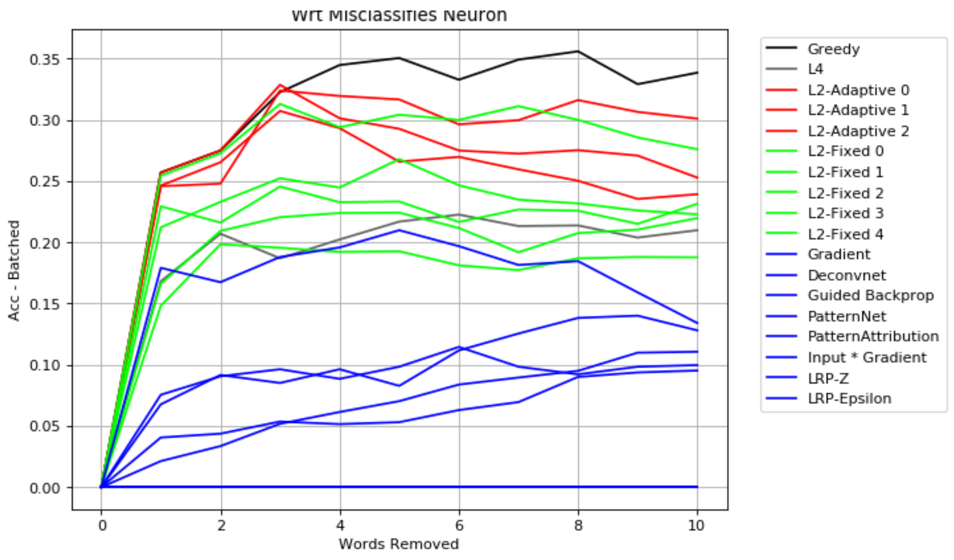
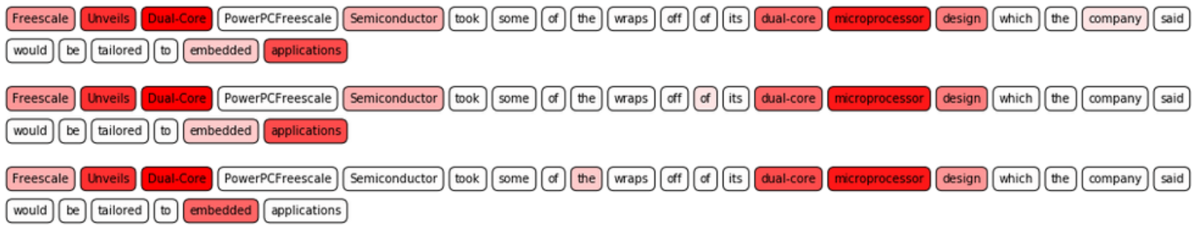


Figure 22: Preposition batched accuracy - Misclassified

6.5.2 AG-News

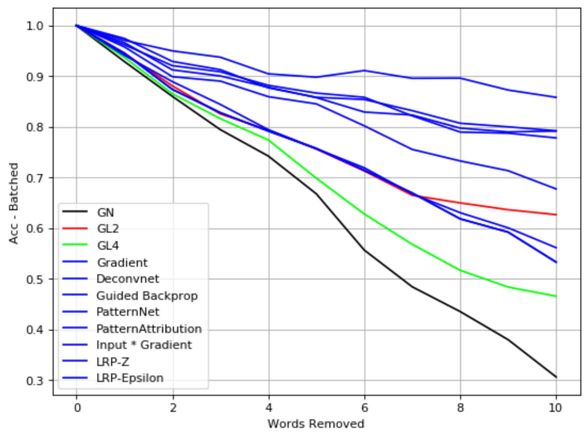

Freescale Unveils Dual-Core PowerPCFreescale Semiconductor took some of the wraps off of its dual-core microprocessor design which the company said would be tailored to embedded applications

Freescale Unveils Dual-Core PowerPCFreescale Semiconductor took some of the wraps off of its dual-core microprocessor design which the company said would be tailored to embedded applications

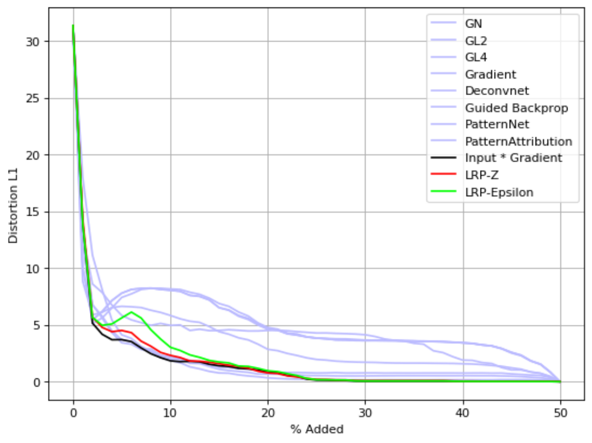
Freescale Unveils Dual-Core PowerPCFreescale Semiconductor took some of the wraps off of its dual-core microprocessor design which the company said would be tailored to embedded applications

Figure 23: Heatmap - text belongs to misc.forsale. Top figure is by vanilla greedy (GN), middle figure by greedy and L_2 (GL2) and bottom figure by greedy and L_4(GL4)

The greedy method and all this variations perform much better than the other methods on this dataset, as evident from the graphs. The greedy method seems to beat the SOTA as such. Compared to [1], the distortion curve for NLP classifiers are much more steep at the beginning so the 'steepness' of RDE/Greedy method can't be seen as such. One can perhaps explore more datasets to do a complete analysis. Perhaps, transformer based classifiers may be less steep as they may depend less on a few particular keywords.



(a) Accuracy with batched prepositions



(b) L1 Distortion

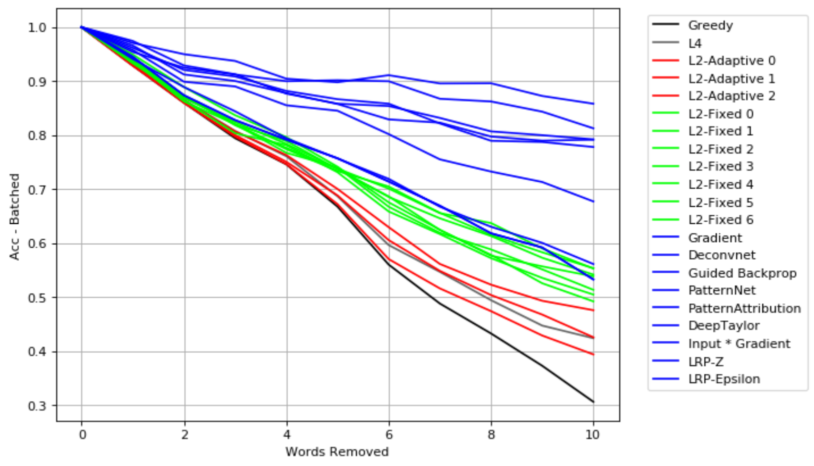


Figure 25: Preposition batched accuracy

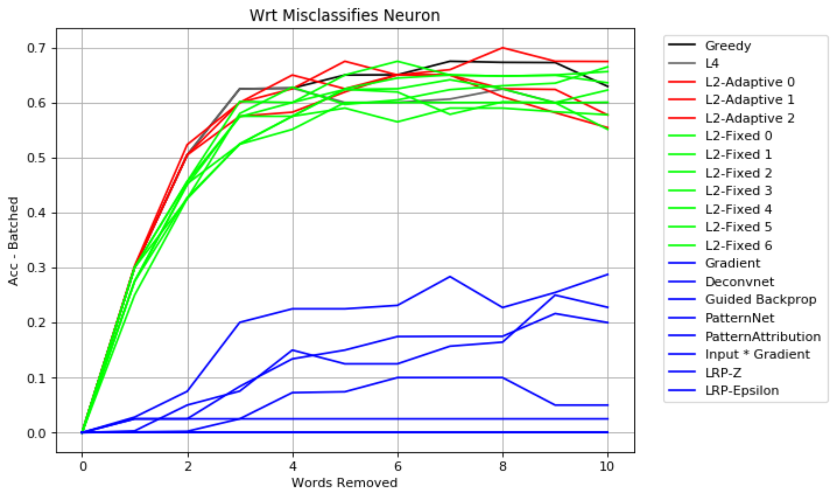


Figure 26: Preposition batched accuracy - Misclassified

7 Knowledge Distillation

7.1 NG20

For the Newgroup-20 model, we tried a student network of lesser complexity. In the student network, we had 8 convolutional kernels/channels instead of 1280 of the teacher network, followed by a dense layer to predict the output (8 to 20 dense layer). Vanilla training on the student led to an accuracy of 58.5% while the teacher had an accuracy of 78.51%. The experiments we tried was based on positive feedback, ie, the words that teacher found to be more relevant were given more importance in the student's training. We took the vanilla trained networks and tried to fine-tune the student network, while keeping the teacher network fixed.

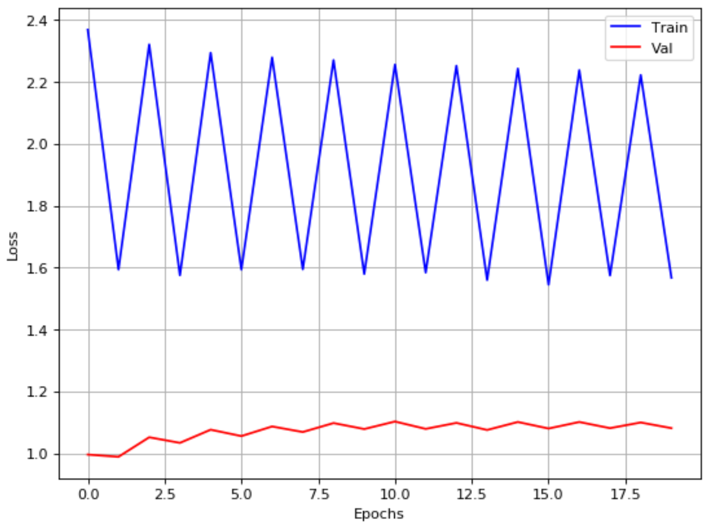


Figure 27: Positive Feedback Training Curve

The main idea is to set the relevance of the unimportant words to be low. 'Important' words can be obtained using any XAI method. we chose LRP to get top 6 most relevant words. In each mini-batch of gradient descent, there were 8 instances of the same document. In 5 of the 8 instances, all unimportant words were set as 0 vectors while in the other 3 instances, they were left unchanged. The important words were unchanged in all 8 instances. This is intuitively equivalent to having just 1 instance of 1024 (the batch size) documents and setting the input of the unimportant words as the original input multiplied by 3/8, ie, it is made closer to 0 vector to tone down it's importance. We did not directly do this as 3/8-th of the original embedding might have drastically different meaning in embedding space!

Let's call the above described training as 'forced' and the normal training process as 'unforced'. In alternated between forced and unforced training in each epoch.

The results are not very promising, the training curve is shown in Figure 27. The student network's accuracy increased only slightly to 59.41% which might have been due to the sort of normalization effect of the feedback training process. The student model might have been overfitted before.

Next, instead of taking 8 instances in a batch, we took a single instance of a document and we set all the unimportant words to 0. This time, we only did forced epochs and we got the student network accuracy as 57.55%. Again, this does not seem to yield satisfactory performance.

7.2 AG News

In the AG News student network, we had two convolutional kernels/channels instead of 1024 of the teacher network, followed by a dense layer to predict the output (2 to 4 dense layer). Vanilla training on the student led to 81.08% accuracy, while the teacher had an accuracy of 91.12%. The experiments we tried was based on both positive and negative feedback.

7.2.1 Positive Feedback

The same experiment described in Section 7.1 was done on the AG News dataset. In a mini-batch, there were 8 instances of 64 unique documents with 5 out of the 8 batches having the unimportant words as zeroed out. We alternated between forced and unforced epochs. The training curve is as shown in Figure 28. The test accuracy of the student decreased to 80.20%. Like the NG20 case, we then took a single instance of a document in each mini-batch and set all the unimportant words to 0 . We alternated between forced and unforced epochs and the student network accuracy became 80.87%. Again, this does not seem to yield satisfactory performance. We tried to do only forced epochs with a reduced learning rate, but the accuracy dropped to 70.88%.

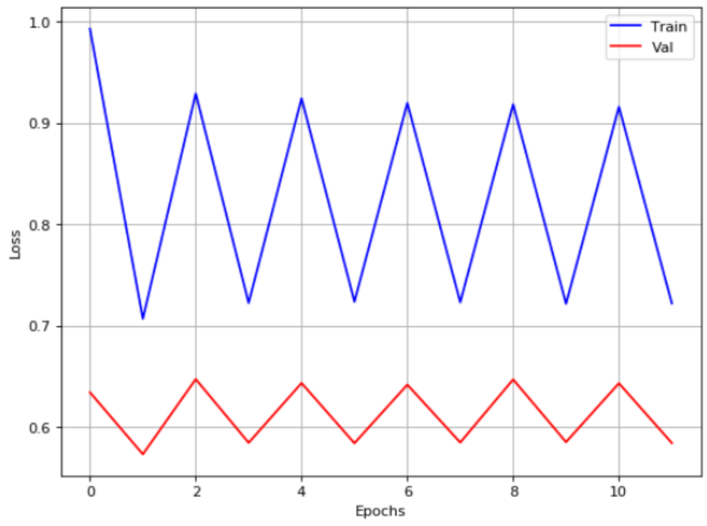
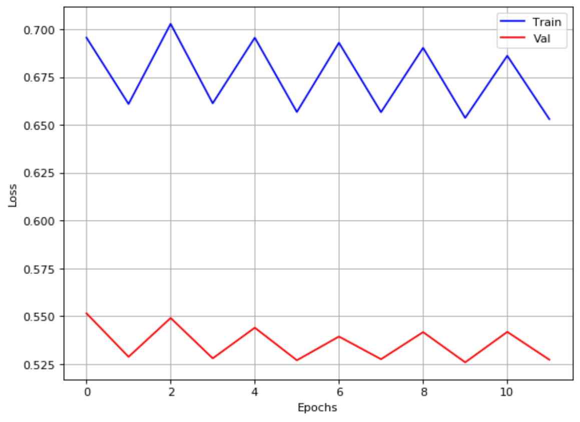


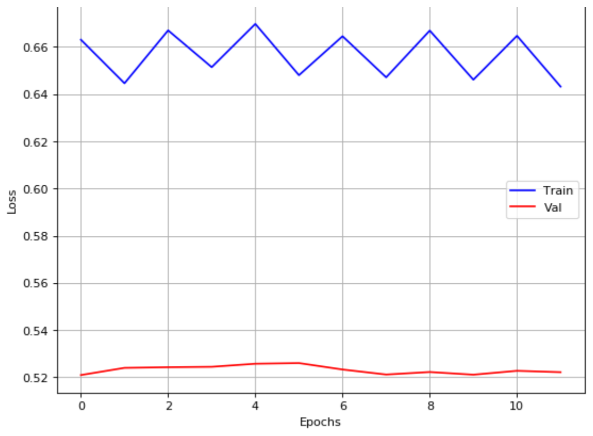
Figure 28: Positive Feedback Training Curve

7.2.2 Negative Feedback

We did the same experiment as the positive feedback experiment with just the small change that in 5 out of the 8 batches, the 'important' words were zeroed out. We alternated between forced and unforced epochs. The training curve is as shown in Figure 29a. The test accuracy of the student decreased to 80.62%. We then took a single instance of a document in each mini-batch and we set all the unimportant words to 0. We alternated between forced and unforced epochs and the student network accuracy became 80.59%, training curve in Figure 29b. Again, this does not seem to yield satisfactory performance.



(a)



(b)

Figure 29: The training curves

7.3 MNIST

Based on the work in [7], we tried to use XAI-based knowledge distillation. Previous works have used soft targets from the teacher model to train the student model. They have used a combination of cross-entropy (CE) loss and KL divergence loss (KLDiv). We describe the previous work in [7] as follows. Let T be the 'temperature'. Let z_i be the logits (pre-softmax values) for teacher model and v_i be the logits for the student model. Since MNIST [8] has 10 classes, z and v are 10-dimensional. The output probabilities of the student model is given by

$$s_i = \frac{e^{v_i}}{\sum_{j=0}^{9} e^{v_j}}$$

The 'soft targets' are obtained by including the temperature in the probabilities. Define p_i and q_i as described.

$$q_i = \frac{e^{v_i/T}}{\sum_{j=0}^{9} e^{v_j/T}}$$

$$p_i = \frac{e^{z_i/T}}{\sum_{j=0}^{9} e^{z_j/T}}$$

The objective loss function is

$$L_1(v, z, y) = \alpha T^2 \cdot KLDiv(p||q) + (1 - \alpha) \cdot CE(s, y)$$

where CE is the cross-entropy loss, y is the ground truth label and α is a hyper-parameter to assign suitable weight to the loss functions.

We believe that we can train the student model better by adding the some extra information for each image. We can 'tell' it how the teacher model focuses on the image for a particular decision. Therefore, can assist the student model in deciding where to focus in a given image. Every model would have different relevance map based on the architecture and the XAI method. However, we think that any sufficiently 'good' XAI method would be helpful for our purpose. We believe that any XAI method which has sufficient discriminative power between the various regions in the image would provide better training ability.

We change the loss function slightly to introduce our XAI-based method. We calculate the RAP based relevance maps for both the teacher and the student. The two relevance maps are of shape $(28, 28)$ each, as every pixel is assigned a relevance value. The relevance values are in range $[0, 1]$. We denote Mean Square Error as MSE. The new loss is as defined

$$L_2(v, z, y, R_s, R_t) = \alpha_k T^2 \cdot KLDiv(p||q) + (1 - \alpha_k - \alpha_m) \cdot CE(s, y) + \alpha_m \cdot MSE(R_s, R_t)$$

where, R_s and R_t are the RAP relevance maps by student model and the teacher model respectively for the same image. α_m, α_k are weight hyper-parameters.

For a given image in a mini-batch, if the teacher model misclassifies it then we do not add the MSE loss corresponding to this sample. We do not want the teacher's relevance map if it can't classify the image! If the teacher model makes the correct prediction, then we consider the RAP map for both the teacher and the student model corresponding to the ground truth class. The overall MSE loss is divided by the number of correct prediction by the teacher model in the mini-batch. The other two losses are divided by the batch size. This is done to compensate for the few misclassified images in the mini-batch.

7.3.1 Teacher Model

The teacher model's architecture is based on 2 fully connected hidden layers, each of size 1200. We used dropout after every layer, and weight decay during training for regularization. The teacher model trained using the conventional method of cross entropy loss attains 99.25% accuracy, i.e., 75 misclassifications on the test set.

7.3.2 Student Model

The student model's architecture is based on 1 fully connected hidden layer of size 800 with no dropout. We did not use weight decay in the optimization process. The results by varying the hyper-parameters are shown in Table 1.

As compared to Experiment-1 (Ex-1), which is the conventional CE loss training, we get better performance with Ex-2, 3, 4, 5. All of these (Ex-2, 3, 4, 5) have the extra KLDiv loss term in the training with a combination of different temperatures and weights. The soft-targets have better 'distilling' or training capability than the simple CE loss, as shown in [7].

Experiment	α_k	α_m	T	Misclassifications
1	0	0	-	132
2	0.75	0	5	108
3	0.25	0	10	112
4	0.25	0	10	112
5	1	0	5	118
6	0	1	-	8800
7	0.5	0.5	10	116
8	0.33	0.33	10	117
9	0	0.5	-	126

Table 1: Results

Ex-6, 7, 8 include our proposed XAI-based method. While they do not perform as well as the previous methods, they still do perform better than vanilla CE loss based training. Ex-6 makes it evident that just RAP maps based MSE loss is a challenging objective function for the student to minimize. This motivates our variational hyper-parameter scheme. We choose to vary the weight hyper-parameter with each epoch. Initially, it is more convenient to train on CE loss than MSE loss, and we can then fine tune on the MSE loss. Hence, we tried the following weight scheme

$$\alpha_m(i) = 1 - 0.5 \times (0.95)^i$$

where, i is the epoch number. We had set $\alpha_k = 0$. Using this hyper-parameter setting, we obtained 140 misclassifications.

There needs to be more rigorous experiments on larger datasets to firmly establish these results. The problem with this experiment is that the student model is comparable to the teacher model even with vanilla CE loss based training. The misclassifications, as a percentage of the total test set, is quite less and hence, the performance does not vary too much to make a concrete claim. We do, however, note that only MSE loss based training is not enough to train the model. We should see it more of a fine-tuning method.

7.4 Places2

We used the pre-trained ResNet50 for Places2 dataset provided by [9] as the teacher model. It achieves a top-1 accuracy of 54.7% on the validation set. Neither the test set annotations were not provided by the authors, nor was the evaluation website functioning properly. Hence, we could only observe the validation set accuracy. The teacher model has about 24 million parameters.
We chose an AlexNet based smaller network with a skip connection as the student model [1]. It had about 3 million parameters. Using simple CE loss training, we obtained a validation accuracy of about 27%

[1]Exact description can be found on Github code

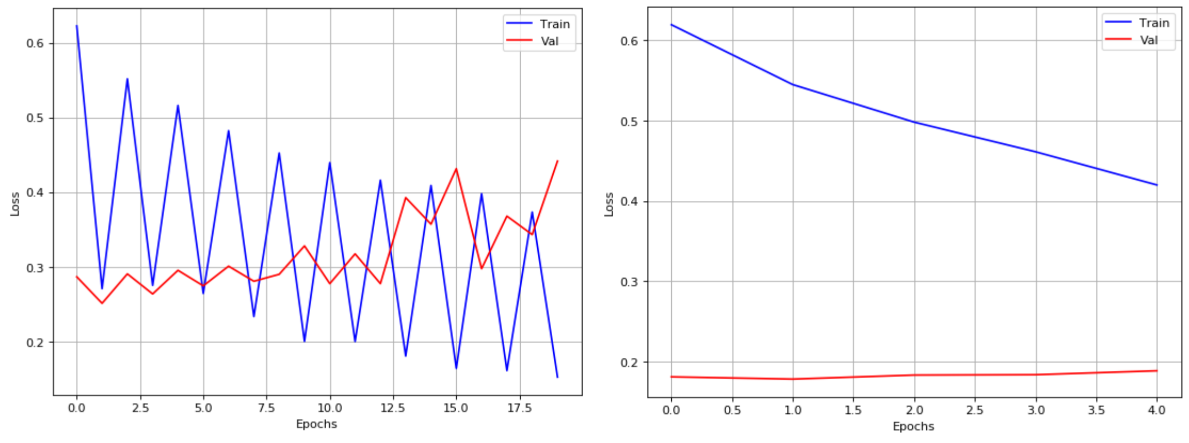
and an accuracy of about 29% when trained with a combination of KLDiv loss and CE loss. We could not implement XAI-based distillation for this dataset due to time constraint. We can, however, confirm from this experiment that the soft-target based training indeed boosts the performance as claimed by [7].

8 Self-Feedback Fine Tuning

The experiments were carried out on AG News dataset. This is to be seen more as of a fine tuning method rather a training method. The model initially has an accuracy of 91.12%. Based on the type of feedback given, we can have both positive and negative feedback.
The idea behind positive feedback is to tell the model that it should focus more on the words that it already considers relevant. This can backfire if the model is not good at the beginning of the fine tuning process. It is possible that it does not place too much confidence on certain keywords but on a few documents, those keywords would help make the correct decision. A positive feedback would help in such situations.
The philosophy behind negative feedback is to force the model to learn more than just keywords. It should be able to do well without a few keywords. In other words, it helps reduce the model bias and might improve performance on the test set.

8.1 Negative Self-Feedback

We removed the top 3 relevant words (by LRP) of each document in the mini-batch and trained on the modified batch. For each batch, we compute the relevance values on the modified network due to the previous gradient step, thus, this is quite time consuming. Similar to the Knowledge Distillation approach, we alternated between the a forced epoch and an unforced epoch (vanilla training). We also did an experiment with only forced epochs. The results are as shown in Figure 30.



(a) Alternate forced and unforced epochs

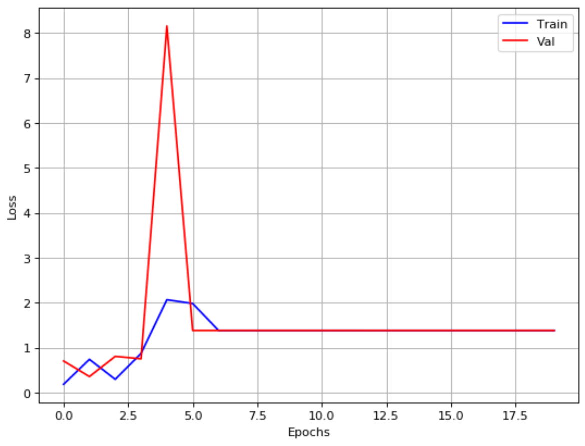
(b) Only forced epochs

Figure 30: The training curves

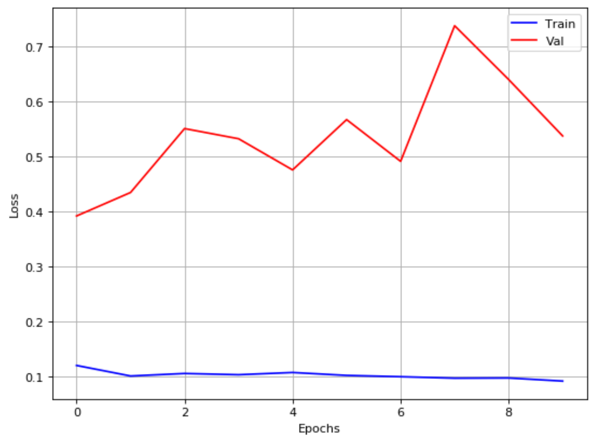
Alternating the methods result in 91.28% accuracy on test set. Only forced epochs result in 91.43% accuracy. Hence, there is a slight increase in the accuracy. However, rigorous experimentation is needed to establish whether negative feedback is indeed beneficial.

8.2 Positive Self-Feedback

This is just the opposite of the negative feedback fine tuning, here we only keep the top 5 words and set the other words as zero vectors. The training curves are as shown.



(a) Alternate forced and unforced epochs



(b) Only forced epochs

Figure 31: The training curves

The testing accuracy achieved by alternating forced and unforced epochs is 25%. It acts like a random classifier and it is clear from the training curve that the training process is unsuccessful. The bad performance can possibly be attributed to a large learning rate. When we tried to do only forced epochs (Figure 31b) with a slightly less learning rate, we achieved an accuracy of 88.54% after the first epoch and 84% after the 10th epoch. On reducing the learning rate to order of 10^{-5}, we achieved 91.7% accuracy within the first 3 epochs.

9 Further Thoughts

Following are topics which can be explored in the future:

- The models described in Section 2 are not as good as the SOTA in NLP which are generally transformer based [10]. RDE method works regardless of the model architecture but other methods like LRP are dependent on the architecture. How to extend LRP to these networks?
- For greedy algorithm described in Section 5, it would be interesting to see it implemented with a small change; we compare the probability values instead of $\Phi(X')_h$,ie, we do the comparison after the softmax layer. This would also incorporate the fluctuations of rest of the neuron activations.
- It would be worthwhile to see the greedy method combined with a fast method like LRP or RAP instead of combining greedy method with L_2 method. It would speed up the algorithm to $O(n)$.
- Negative feedback distillation described in Section 7.2.2 may be useful in domain adaption problems where we want to get rid of training dataset bias.
- In sections 7.1, 7.2 and 8, we could try to generate a noise such that the resultant sample point lies of the training data space. Replacing the words by zero vectors creates a sample from a space which the network has not seen. If we could train an inpainting GAN to 'inpaint' the words, then the results may improve.

- In Section 7.3, we can slightly change the XAI-based MSE loss. Instead of calculating the MSE loss between the RAP maps of the ground truth label, we can take the MSE loss of the all the 10 map pairs generated by considering all the 10 output labels. This further teaches the student why the rest 9 classes are wrong.

References

[1] Jan MacDonald et al. "A Rate-Distortion Framework for Explaining Neural Network Decisions". In: *CoRR* abs/1905.11092 (2019). arXiv: `1905.11092`. URL: `http://arxiv.org/abs/1905.11092`.

[2] Leila Arras et al. "Explaining Predictions of Non-Linear Classifiers in NLP". In: (2016), pp. 1–7.

[3] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: (Jan. 2013), pp. 1–12.

[4] Ronan Collobert et al. "Natural Language Processing (almost) from Scratch". In: *CoRR* abs/1103.0398 (2011). arXiv: `1103.0398`. URL: `http://arxiv.org/abs/1103.0398`.

[5] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: (Oct. 2014), pp. 1746–1751. DOI: `10.3115/v1/D14-1181`. URL: `https://www.aclweb.org/anthology/D14-1181`.

[6] Armand Joulin et al. "FastText.zip: Compressing text classification models". In: *arXiv preprint arXiv:1612.03651* (2016).

[7] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *ArXiv* abs/1503.02531 (2015).

[8] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.

[9] Bolei Zhou et al. "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[10] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).