Semi-Supervised Learning for Vision-and-Language Tasks using MixMatch

Kalpesh Krishna
kalpesh@cs.umass.edu

Videsh Suman
vsuman@cs.umass.edu

Abstract

Semi-supervised learning algorithms attempt to train a model with limited labeled data by leveraging a large amount of unlabelled samples . However, there is limited literature on using such a strategy for visual-language tasks, mainly due to the complexity and the discreteness of the input space. We attempt to reformulate `MixMatch`, a recently proposed semi-supervised learning strategy, on a state-of-the-art multi-modal framework `LXMERT`, and report the performance on the NLVR2 dataset. We compare these results with suitable baseline experiments. To assess the applicability of textual interpolation, we conduct an interesting experiment on GPT-2. Towards the end, we propose two more modifications that were planned but couldn't be executed due to the constraint of time.

1. Introduction

Deep neural networks, while quite successful, are data hungry in supervised settings. They typically require large hand-curated datasets where domain experts label the mapping from the input to the output. The large scale of the datasets is critical to train deep neural networks due to the large number of learnable parameters. Building these large datasets is expensive and often impractical.

To tackle this issue, semi-supervised learning (SSL) [3] algorithms have been developed, which use large amounts of unlabelled data, in addition to some labelled data, to learn meaningful semantics in the input space. Very recently, a holistic SSL strategy `MixMatch` [2] has been shown to significantly improve the image classification performance in semi-supervised settings. `MixMatch` is based on three key SSL paradigms — 1) data augmentation via input perturbation; 2) guessing low-entropy labels for data augmented unlabeled examples; 3) linear interpolation of labeled and unlabeled examples in both input and output space using `mixup` [24].

While MixMatch works well for image classification, it is non-trivial to extend this algorithm to more complex input spaces (like multi-modal vision + language), and more complex output spaces (beyond classification like structured prediction or text generation). For these tasks, each of the three SSL principles used by MixMatch cannot be used in their original form — 1) input perturbation is hard for textual inputs since it is important to preserve a grammatically valid input; 2) for text generation tasks like Visual Question Answering [1] entropy minimization is non-trivial since the output space is discrete and unbounded; 3) text cannot be linearly interpolated due to the discrete nature of the input space.

In this work, we attempt to reformulate the MixMatch pipeline for multi-modal vision+language tasks. We focus on a recently proposed vision-and-language cross-modality framework `LXMERT` [17]. To tackle the above mentioned issues, we conduct the following experiments — `mixup` interpolation of input sentences in latent space through a modified implementation of [15] on GPT2 [12]; `mixup` interpolation of the encoded cross-modal features; the MixMatch pipeline incorporating these modifications. We leave two important ideas for future — perturbation of the visual and textual embeddings by adding noise; moving the `mixup` interpolation step to the embedding space before the `LXMERT` encoders. We have open-sourced our current implementation.[1]

2. Related Work

There is a large body of literature on semi-supervised learning methods, however we focus only on some of the more recent ideas which `MixMatch` builds upon. Towards the end of this section, we also discuss some of the advances in building `BERT`-style multi-modal frameworks that have improved the state-of-the-art on most vision-and-language tasks by a significant margin compared to the previous methods.

2.1. Paradigms of Semi-Supervised Learning

Many recent approaches add an SSL loss term which is computed on unlabeled data and seeks to induce generalization towards unseen data. In much of the recent work, this loss term belongs to one of the three classes: consistency regularization — which encourages the model to produce

[1] `https://github.com/martiansideofthemoon/mixmatch_lxmert`

a similar output distribution with perturbed inputs; entropy minimization [7] — which encourages the model to make confident predictions on unlabeled data; and generic regularization —– which encourages the model to generalize well on the unseen data and avoid overfitting the training data.

2.1.1 Consistency Regularization

A very common practice in supervised machine learning pipelines is *data augmentation*, which seeks to transform the datapoints in their input space, keeping the output labels unaffected. Such a regularization technique can artificially expand the size of the training set, and induce a more robust learning to the model. In the context of SSL, data augmentation promotes consistency regularization by leveraging the idea that a classifier should output the same class distribution for an unlabeled example even after it has been augmented. More formally, consistency regularization enforces that an unlabeled sample x should be classified the same as any augmentation of itself $aug(x)$. For example, in image classification, it is common to add noise or elastically deform an input image, that can significantly change the overall pixel content without change its label. Consider a generic model $p_m(y|x;\theta)$ that obtains a distribution over class labels y for an input x and parameters θ. In a simplistic case of SSL, for the unlabeled datapoints x, [14] adds the loss term,

$$||p_m(y \mid aug(x);\theta) - p_m(y \mid aug(x);\theta)||_2^2 \qquad (1)$$

Note that the two terms in eq. (1) are not identical as $aug(x)$ is a stochastic transformation. `Mean Teacher` [18] replaces one of the terms in eq. (1) with the output of the model using an exponential moving average of model parameters. This approach has been empirically found to significantly improve the results. The original `MixMatch` [2] approach utilizes a form of consistency regularization through the use of standard data augmentation for images (random horizontal flips and crops). With the input space being discrete and complex in the case of vision-and-language datasets, it's non-trivial to apply the data augmentation.

2.1.2 Entropy Minimization

For more confident predictions, a common hypothesis is that the classifier's decision boundaries should remain within the low-density regions of the marginal data distribution. [7] suggested to use a loss term which minimizes the entropy of the predicted distribution $p_m(y \mid x;\theta)$ on unlabeled data x. `MixMatch` [2] implicitly achieves entropy minimization through the use of a "sharpening" function on the target distribution over augmented sets of unlabeled data. Since we omit the data augmentation step in our method, the entropy minimization is achieved by constructing hard (1-hot) labels from guessed predictions on unlabeled data. `Pseudo-Label` [10] suggests a similar approach of hard-labeling the high-confident predictions on unlabeled data and using them as training targets in a standard cross-entropy loss.

2.1.3 Traditional Regularization

Regularization refers to the general approach of imposing a constraint on a model to make it harder to memorize the training data, and therefore hopefully make it generalize better to unseen data [8]. `MixMatch` uses weight-decay by penalizing the L_2 norm of the model parameters. The most interesting part of `MixMatch` is the utilization of `mixup` as both a regularizer (applied to labeled datapoints) and a semi-supervised learning method (applied to unlabeled datapoints). This, in some sense encourages convex behavior "between" the examples. For our experiments, we leverage `mixup` on the NLVR^2 [16] task in a similar way.

2.2. Vision-and-Language Frameworks

After the recent success of BERT [5], a number of multi-modal frameworks [17], [11], [4] have been proposed that leverage large-scale pre-training with masked objectives similar to language modeling. These models perform significantly better than the previous state-of-the-art methods on several language-and-vision tasks. These frameworks serve as proxies for semi-supervised learning, allowing sample efficient learning of downstream tasks [21]. In this project, we use the pre-trained `LXMERT` (Figure 1) framework for all our experiments. `LXMERT` is essentially a large scale Transformer [19] model, consisting of three encoders — an object relationship encoder, a language encoder, and a cross-modality encoder. For our SSL experiments, we finetune `LXMERT` on NLVR^2 [16] with limited labels.

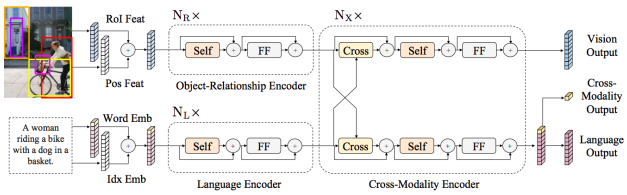

Figure 1. (Source: Tan et al. [17]) For our experiments, we have used the pre-trained LXMERT model. This figure shows the pipeline (including the 3 encoders) for learning vision-and-language cross-modality representations. 'Self' and 'Cross' are abbreviations for self-attention sub-layers and cross-attention sub-layers, respectively. 'FF' denotes a feed-forward sub-layer.

3. Method

In this section, we discuss the vanilla `MixMatch` [2] approach in detail. This method obtained state-of-the-art SSL results by a significant margin at the time of release. We then go on to describe the challenges of using `MixMatch` on vision-and-language tasks as is, and then go on to propose our modifications, most of which we have implemented as a part of this project.

3.1. The Original MixMatch Algorithm

In their paper, Berthelot et al. refer to `MixMatch` as being a "holistic" framework that incorporates the ideas and components from the dominant paradigms of SSL (discussed in Section 2.1).

3.1.1 Data Augmentation

Typical to most SSL methods, `MixMatch` also involves data augmentation both on labeled and unlabeled data. For each x_b in the batch of labeled data $\mathcal{X}$, a transformed version $\hat{x}_b = aug(x_b)$ is generated. For each u_b in the batch of labeled data $\mathcal{U}$, K such transformations $\hat{u}_{b,k} = aug(u_b)$ $(k \in (1, ..., K))$ are generated. For an image classification task, it maybe trivial, but for tasks with textual or multimodal inputs, this step is hard owing to their discreteness.

3.1.2 Label Guessing

For each unlabeled example in $\mathcal{U}$, `MixMatch` produces a "guess" for the example's label using the model's predictions. This guess is later used in the unsupervised loss term. To do so, the average of the model's predicted class distributions across all the K augmentations of u_b by

$$\overline{q}_b = \frac{1}{K} \sum_{k=1}^{K} p_m(y \mid \hat{u}_{b,k}; \theta) \tag{2}$$

This enforces consistency regularization amongst the augmentations of unlabeled examples. Over this averaged class prediction, an additional step of sharpening is applied to induce low-entropy guessing. A common approach of adjusting the "temperature" of the categorical distribution can be defined by,

$$Sharpen(p, T)_i = \frac{p_i^{1/T}}{\sum_{j=1}^{L} p_j^{1/T}} \tag{3}$$

where p is an input categorical distribution and T is a hyperparameter. In the context of `MixMatch`, they use $q_b = Sharpen(\overline{q}_b, T)$ as the target distribution for the model prediction for any $u_{b,k}$ an augmentation of u_b. It can be seen that lowering the temperature T induces low-entropy predictions by the model. Since we have omitted the data augmentation step in our current implementation, we enforce hard labeling over the guessed predictions.

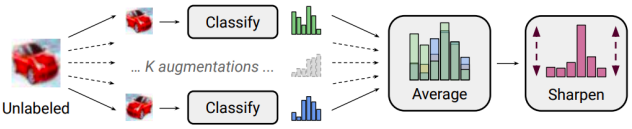

Figure 2. (Source: Berthelot et al. [2]) Diagram of the label guessing process proposed in original `MixMatch`. They applied stochastic data augmentation to each unlabeled image K times, and each augmented image was fed through the classifier. Then, the average of these K predictions was "sharpened" by adjusting the distribution's temperature T.

3.1.3 `mixup` Interpolation

The authors modify the originally proposed `mixup` [24] regularization method, and use it to mix both labeled samples and unlabeled samples with label guesses. A pair of two examples (x_1, p_1) and (x_2, p_2), a "mixed-up" sample can be computed by,

$$\lambda \sim Beta(\alpha, \alpha) \tag{4}$$
$$\lambda' = max(\lambda, 1 - \lambda) \tag{5}$$
$$x' = \lambda' x_1 + (1 - \lambda') x_2 \tag{6}$$
$$p' = \lambda' p_1 + (1 - \lambda') p_2 \tag{7}$$

where α is a hyperparameter. Through (5), it is ensured that x' is closer to x_1 than x_2.

To apply the `mixup` interpolation, it is important to collect the augmented labeled examples and the augmented unlabeled examples with guessed labels in separate collections.

$$\hat{\mathcal{X}} = ((\hat{x}_b, p_b); b \in (1, ..., B)) \tag{8}$$
$$\hat{\mathcal{U}} = ((\hat{x}_{b,k}, p_b); b \in (1, ..., B), k \in (1, ..., K)) \tag{9}$$

To create a data source for `mixup`, is formed by combining and shuffling $\hat{\mathcal{X}}$ and $\hat{\mathcal{U}}$. Now, for each i^{th} sample in $\hat{\mathcal{X}}$, $\texttt{mixup}(\hat{\mathcal{X}}_i, \hat{\mathcal{W}})_i$ is computed and added to a new collection $\hat{\mathcal{X}}'$. Then, $\hat{\mathcal{U}}'$ is populated with $\texttt{mixup}(\hat{\mathcal{U}}_i, \hat{\mathcal{W}}_{i+|\hat{\mathcal{X}}|})$ for $i \in (1, ..., |\hat{\mathcal{U}}|)$, using the remainder of $\hat{\mathcal{W}}$. In summary, the strategy is to transform $\hat{\mathcal{X}}$ into $\hat{\mathcal{X}}'$, a collection of artificial labeled samples having undergone data augmentation and `mixup`(potentially with an unlabeled sample); and $\hat{\mathcal{U}}$ into $\hat{\mathcal{U}}'$, a collection of multiple augmentations of each unlabeled sample with corresponding labeled guesses and `mixup`. In contrast, we simply `mixup` the samples directly from collections $\mathcal{X}$ and $\mathcal{U}$, dropping the data augmentation step.

3.1.4 Loss Function

Samples from these processed batches $\hat{\mathcal{X}}'$ and $\hat{\mathcal{U}}'$ are used to compute semi-supervised loss,

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\hat{\mathcal{X}}'|} \sum_{x,p \in \hat{\mathcal{X}}'} H(p, p_m(y \mid x; \theta)) \tag{10}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\hat{\mathcal{U}}'|} \sum_{u,q \in \hat{\mathcal{U}}'} ||q - p_m(y \mid x; \theta)||_2^2 \tag{11}$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}} \tag{12}$$

where L is the number of unique class labels, $H(p, q)$ is the cross-entropy loss between the distributions p and q, and $\lambda_{\mathcal{U}}$ is a hyperparameter. The authors claim that L_2 distance between the predicted and the guessed labels of unlabeled mixed-up examples keeps the loss term bounded and less sensitive to incorrect predictions. For our experiments, we do not alter the loss terms.

3.2. MixMatch on Vision+Language Tasks

Vision-and-language tasks require an understanding of visual concepts, language semantics, and, most importantly, the alignment and relationships between these two modalities. Most tasks like VQA [6], NLVR2 [16], VCR [22] have a complex multi-modal input space containing text as well image(s). This discreteness in the input space makes it difficult to extend the existing semi-supervised learning approaches for these multi-modal tasks. Other text generation tasks like image captioning involve a discrete and unbounded output space which is harder to deal with in a limited labeled data setting. From the discussion in Section 3.1 and the results in [2], we conclude that `MixMatch` is a promising SSL strategy for image classification task, though there are significant issues to apply this approach to a task on vision-and-language. In the following subsections, we discuss the challenges in extending `MixMatch` on a variety of multi-modal tasks. However, in the interest of time, we restrict our experiments just to the NLVR2 dataset. Figure 3 gives a summary of the task through two examples.

3.2.1 Issues

From what we discussed earlier (Section 3.1), MixMatch works so well because of its holistic approach towards incorporating the three dominant principles of SSL. Although it's hard to extend all of these three principles for these vision -and-language tasks:

- **Consistency Regularization:** As discussed earlier, *data augmentation* is one of the most common approaches to enforce consistency regularization in the training set. For tasks with textual inputs, it is non-trivial to generate grammatically valid augmentations due to the discrete nature of text. The common techniques accounting for this—paraphrase generation using back-translation, word replacement [20]—are unreliable since they distort semantics or lead to grammatically invalid inputs.

  Another roadblock in adopting `MixMatch` to `LXMERT` (our choice of framework) is the discrete visual input space. `LXMERT` uses pre-trained Faster R-CNN [13] features of the input images. Faster R-CNN features are generally unaffected by perturbations of the input space (since image semantics are preserved) removing the benefits of consistency regularization.

- **Entropy Minimization:** For a task that is not classification (i.e. involving structured prediction or text generation), it's hard to guess the predictions and average these predictions over the augmented datasets. This is because the output space has an exponentially large support making the computation of the normalization constant in marginals intractable. For instance, in image captioning models it is computationally infeasible to compute the full marginal over captions.[2]

- **`mixup` Interpolation:** It is not possible to perform

Figure 3. (Source: Suhr et al. [16]) Two examples from NLVR2. Each caption along with a pair of associated images comprise of the input space. The task is to predict if the caption is True or False. The examples require addressing challenging semantic phenomena, including resolving *twice ... as* to counting and comparison of objects, and composing cardinality constraints, such as *at least two dogs in total* and *exactly two*.

[2]While it is possible to sample sequential models without replacement [9], it's unclear how these samples could be used to achieve entropy minimization.

mixup on discrete input spaces since it requires a linear interpolation of the inputs which will likely result in points outside the support of the input space. In Vision+Language tasks, the input space has text, which is a discrete object. Similar to the issues under **Consistency Regularization**, Faster-RCNN feature inputs (in place of continuous images, are also discrete and cannot be directly used for mixup interpolation.

3.2.2 Proposed Solutions

In this section we propose solutions to each of the issues discussed in the previous section, and describe how we extend MixMatch to Vision+Language tasks. Note that we did not have sufficient time to complete the proposed modification for Consistency Regularization and avoided this regularization in our MixMatch implementation.

- **Consistency Regularization:** Discrete objects such as words are typically fed into deep neural networks by dense embedding vectors. Inspired by [23], we propose augmenting the dataset with consistent labels by adding gaussian noise to the dense embedding vectors of the discrete objects.

- **Entropy Minimization:** We use only the argmax prediction (hard labeling) to minimize entropy. Since we focus on a binary classification task (NLVR2), entropy minimization is not problematic.

- **mixup Interpolation:** To perform mixup on textual input spaces, we experiment with two strategies. First, we qualitatively analyze the outputs of a pretrained text generation model GPT-2 [12] conditioned on the latent space interpolation of two sentences (using a recent technique for inducing latent spaces onto pretrained language models [15]). After noticing only lexical overlaps in the interpolated outputs and no semantic mixture, we perform mixup on the dense encoded LXMERT representations rather than the inputs themselves.

4. Experiments

In this section we describe the experimental details and the associated findings.

4.1. Interpolating Text using GPT-2

We conduct a preliminary experiment to assess the feasibility of interpolating sentences. As discussed in the previous section, this is required for the mixup step of the MixMatch algorithm.

Setup: We leverage a recently introduced technique [15] to create a bijective mapping between sequences and vectors in a latent space, leveraging the knowledge of a pre-trained language model. We slightly modify the algorithm[3] to make it compatible with the pre-trained state-of-the-art language model GPT-2 [12]. With this setup, we map two sentences into the latent space and map their interpolation back into the sequence space. To improve the generation quality, we fine-tune GPT-2 on the NLVR2 training set inputs before interpolation.

Results: We present qualitative results in Table 1. Since the model is built using a fluent language model GPT-2, interpolations tend to be grammatically valid (even if not pragmatic / meaningful). As evident from the table, interpolations tend to be lexical (having word overlaps) rather than semantic (mixture of the meaning of the two sentences). This encourages the interpolation of the dense representations of the text inputs rather than the sentences themselves during mixup.

4.2. Cross-Modal MixMatch on LXMERT

We conduct a series of experiments on the pretrained LXMERT model with NLVR2 dataset. With each experiment, we move a step closer to our proposed version of the MixMatch technique.

Setup: The details of fine-tuning LXMERT with the NLVR2 dataset have been mentioned in their paper [17]. We use LXMERT to encode the two image-statement pairs $(img0, s)$ and $(img1, s)$ for each example, where $img0$ and $img1$ are the object detection feature vectors (from pretrained Faster R-CNN [13]) of the two images and s is the associated textual caption. Then, we n train a classifier based on the concatenation of the two cross-modality outputs in various limited training data settings.

Partial Data Baselines: To set baselines for any semi-supervised learning strategy, we perform this naive initial experiment with limited training data and report the results in Table 2.

Vanilla Self-Training: We use the model, learned from the labeled examples, to make hard guesses on the unlabeled examples. We, then, re-use these unlabeled samples for training this model further. We do this in two different ways — 1) **batch-wise:** training on all the labeled examples for multiple epochs, then, making guesses for all the unlabeled examples and finally, re-training the model with both collections of examples from the pre-trained check-

[3] Instead of adding the latent vectors to the hidden representations of the language model, we add them to the input embeddings of the GPT-2 transformer.

λ value	Example 1	Example 2
0.0	At least one panda is playing with a bubble.	There is exactly one sink in one of the images.
0.2	At least one panda is playing with a toy gun.	There are two dogs in the right image.
0.4	At least one dog is standing on the side of green grass.	There are two dogs in the image on the right.
0.6	A single white building with a chimney is on the left side of white roof.	At least two round tables are at the bar in the image on the right.
0.8	A red chimney rises from a white chimney.	At least two round plates are v in the image on the right.
1.0	A red chimney rises from a yellow building with a thatched roof.	At least two round plates are clearly visible in the image on the right.

Table 1. Interpolation between sentences from the NLVR2 dataset. This was done by generating sentences from a latent space induced over a pretrained language model [15]. Notice that the interpolations, while grammatically correct, tend to show lexical overlap instead of semantic overlap.

Fraction	10%	25%	50%	100%
Performance	61−63%	67−69%	71.4%	74.4%

Table 2. Training the model with limited labeled data only. **Fraction** refers to amount of the NLVR2 training set used, while **Performance** refers to the classification test accuracy.

Fraction Labeled	10%	25%	50%
Performance (batch-wise)	58.2%	62.1%	−
Performance (iterative)	64.2%	68.4%	70.7%

Table 3. Training the model using vanilla self-training. **Fraction** refers to amount of labeled samples used from the NLVR2 training set. The rest were used as unlabeled examples. The batch-wise training experiment for 50% did not converge.

Fraction	10%	25%	50%	100%
Performance	60.5%	61.4%	71.9%	74.1%

Table 4. Training the model with only the "`mixed-up`" labeled examples in the limited data setting.

point; 2) **iterative:** in each training iteration, learn from the labeled mini-batch $\mathcal{X}_{mini}$, guess hard labels for the unlabeled mini-batch $\mathcal{U}_{mini}$, and re-learn from the combined mini-batch $(\mathcal{X}+\mathcal{U})_{mini}$. We report the results in Table 3.

`mixup` Regularization on Labeled Data: We performed the initial experiment of using limited training data with `mixup` interpolation between pairs of labeled examples. Every mini-batch is interpolated with a shuffled version of itself with hope of better generalization. We tune the hyperparameter α for this experiment. Table 4 contains the results.

The MixMatch Pipeline: Ignoring the data augmentation step, we introduce some modifications to the original `MixMatch` approach. A brief summary of the pipeline — the model takes in mini-batches of labeled and unlabeled data; performs label-guessing and sharpening (rounding the soft probabilities) on the unlabeled mini-batch; creates a new mini-batch after shuffling and combining the two mini-batches of labeled data and unlabeled data with guessed labels; performs `mixup` as described in 3.1.3, to produce the transformed mini-batches, that eventually participate in learning using the semi-supervised loss function (12). We tune hyperparameters α and $\lambda_{\mathcal{U}}$ for this experiment. Refer to Table 5 for results.

Fraction Labeled	10%	25%	50%
Performance (iterative)	62.9%	65.7%	71.7%

Table 5. Training the model with the proposed `MixMatch` approach. As proposed originally, we apply `MixMatch` in the iterative setting only.

5. Conclusion

The results of the experiment with GPT-2, though interesting, do not encourage using it as a substitute for textual interpolation in the input space. The subsequent experiments towards deploying SSL on `LXMERT`, don't seem produce consistent results. It's astonishing that self-training performances (Table 3) are worse (or about the same) when compared to the partial data baselines (Table 2). This directs us to think about the training stability of the pretrained `LXMERT` framework, especially when trained with lower fractions of labeled data. Even using the vanilla `mixup` on labeled examples (Table 4) does not improve the performance when compared to the baselines. The hope of generalization is not fulfilled as far as this dataset is concerned.

Training with our flagship approach (Table 5) produces

even worse results when compared to the vanilla self-training (iterative) approach. Though we did not employ any data augmentation step with it, the bigger worry is the poor performance of the vanilla self-training method. Nevertheless, we have looked into some other modifications that could improve the performance of our `MixMatch`.

6. Future Work

Due to the constraints on time, we couldn't experiment with other modifications as planned. Here, we list down two major ideas as possible directions for future work.

- As a means to encourage consistency regularization, we propose to augment both the visual and textual embeddings (see Figure 1 for each input by adding noise [23]. Since we finetune the pretrained `LXMERT` for our experiments, this embedding space should represent the inputs suitably well. Gaussian distribution, being a more natural choice for noise sampling, can be introduced this way,

$$X_{emb} \leftarrow X_{emb} \odot e, \quad e \sim \mathcal{N}(I, \sigma^2 I) \tag{13}$$

  where X_emb is an embedding vector, e is a noise matrix and $\odot$ is the element-wise multiplication.

- Currently, the model performs `mixup` computation on the cross-modality responses (Figure 1) of LXMERT. We plan to move this computation back to the embedding space (Figure 4), which is also where we propose to perform the data augmentation step. The hope is that LXMERT encoders will learn better representations with the transformed input embeddings.

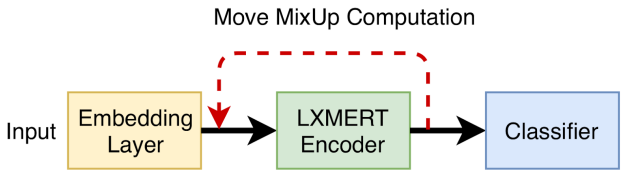

Figure 4. A modification in the current approach, can be experimented with in future.

References

[1] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

[2] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

[3] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 2009.

[4] Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

[7] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2005.

[8] G. E. Hinton and D. van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *COLT*, 1993.

[9] W. Kool, H. Van Hoof, and M. Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. *arXiv preprint arXiv:1903.06059*, 2019.

[10] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 2013.

[11] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.

[12] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[13] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[14] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NIPS*, 2016.

[15] N. Subramani, S. Bowman, and K. Cho. Can unconditional language models recover arbitrary sentences? In *NeurIPS*, 2019.

[16] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi. A corpus for reasoning about natural language grounded in photographs. *arXiv preprint arXiv:1811.00491*, 2018.

[17] H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *EMNLP*, 2019.

[18] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[20] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*, 2019.

[21] D. Yogatama, C. d. M. d'Autume, J. Connor, T. Kocisky, M. Chrzanowski, L. Kong, A. Lazaridou, W. Ling, L. Yu, C. Dyer, et al. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*, 2019.

[22] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi. From recognition to cognition: Visual commonsense reasoning. In *CVPR*, 2019.

[23] D. Zhang and Z. Yang. Word embedding perturbation for sentence classification. *arXiv preprint arXiv:1804.08166*, 2018.

[24] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.