[Re] On Adversarial Mixup Resynthesis

Narayanan Elavathur Ranganatha
Department of Computer Science and Engineering
Manipal Academy of Higher Education
Manipal, Karnataka 576104
naruarjun@gmail.com

Aniket Didolkar
Department of Computer Science and Engineering
Manipal Academy of Higher Education
Manipal, Karnataka 576104
adidolkar123@gmail.com

Abstract

In this paper we aim to reproduce the experiments of Beckham *et al.* [2019]. Beckham *et al.* [2019] introduced a new data augmentation technique. They propose a method for interpolating hidden state representations of images to produce an image that belongs to one of the class labels in the dataset. They use adverserial loss for training this system. The feature representations obtained through this approach is tested on classification tasks on various datasets. The datasets used for classification include MNIST, KMNIST and SVHN.

1 Introduction

Mixup is a data augmentation technique. Data augmentation greatly helps in improving generalization performance. A practical intuition for why this is the case is that by generating additional samples,we are training our model on a set of examples that better covers those in the test set.

Mixup Zhang *et al.* [2017] is a regularisation technique which encourages deep neural networks to behave linearly between pairs of data points. These methods artificially augment the training set by producing random convex combinations between pairs of examples and their corresponding labels and training the network on these combinations. This has the effect of creating smoother decision boundaries, which was shown to have a positive effect on generalisation performance. Arguably however, the downside of mixup is that these random convex combinations between images may not look realistic due to the interpolations being performed on a per-pixel level.

This paper explores mixup in the context of unsupervised learning. The authors propose to perform mixup on the latent space representations of the images. The trained model is used for downstream tasks where the models weights are frozen. These downstream tasks include classification on 3 datasets - *MNIST*, *KMNIST*, *SVHN*.

2 Methodology

The model introduced by the authors consists of an auto-encoder, a mixer and a discriminator. The primary aim of the auto-encoder is to learn representations of the input data that are useful Bengio [2011]. The reconstruction loss is defined as equation 1.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

$$\min_F \mathbb{E}_{x\sim p(x)}||x - g(f(x))||_2 \tag{1}$$

The authors use an adversarial auto-encoder Makhzani *et al.* [2015] with the adversary placed on the reconstruction and the discriminator tries to distinguish between the reconstruction and the real images as this leads to better reconstruction by the auto-encoder. This leads to the following minimization objective:

$$\begin{aligned} \min_F \mathbb{E}_{x\sim f(x)}\lambda||x - g(f(x))||_2 + l_{GAN}(D(g(f(x))), 1) \\ \min_D \mathbb{E}_{x\sim f(x)} l_{GAN}(D(x), 1) + l_{GAN}(D(g(f(x)), 0) \end{aligned} \tag{2}$$

here l_{GAN} is a GAN-specific loss function. In our case, l_{GAN} is the binary cross-entropy loss,which is similar to Goodfellow *et al.* [2014].

The mixer $Mix(h_1, h_2)$ is used to perform combinations of the latent representations of a pair of inputs. Latent representations are generated by passing the image through the encoder $f(.)$. The auto-encoder tries to generate latent representations that when mixed and passed through the decoder $g(.)$, generates images indistinguishable from real images. Finally the minimization objective generated is used to encourage the reconstruction of both the original latent representations and the mixed latent representations to be as realistic as possible. The culmination of all these factors leads to the following loss function for the training of the auto-encoder and the discriminator:

$$\begin{aligned} \min_F \mathbb{E}_{x,x'\sim p(x)} \underbrace{\lambda||x - g(f(x))||_2}_{\text{reconstruction}} + \underbrace{l_{GAN}(D(g(f(x))), 1)}_{\text{fool D with reconstruction}} + \underbrace{l_{GAN}(D(g(Mix(f(x), f(x')))), 1)}_{\text{fool D with mixes}} \\ \min_D \mathbb{E}_{x,x'\sim p(x)} \underbrace{l_{GAN}(D(x), 1)}_{\text{label x as real}} + \underbrace{l_{GAN}(D(g(f(x))), 0)}_{\text{label reconstruction as fake}} + \underbrace{l_{GAN}(D(g(Mix(f(x), f(x')))), 0)}_{\text{label mixes as fake}} \end{aligned} \tag{3}$$

The authors have focused on two ways to combine the latent representations. Manifold Mixup Verma *et al.* [2018] implements mixing in the hidden space through convex combinations:

$$Mix_{mixup}(h_1, h_2) = \alpha h_1 + (1 - \alpha)h_2 \tag{4}$$

where $\alpha \in [0, 1]$ is sampled from a Uniform(0,1) distribution. We can interpret this as interpolating along line segments, as shown in Figure 1.

The authors also explore a strategy in which some components of the hidden representation are randomly retained from h_1 and use the rest from h_2, where h_1 and h_2 are latent representations of 2 images obtained from the encoder and in this case we would randomly sample a binary $mask \in \{0, 1\}^k$ (where k denotes the number of feature maps) and perform the following operation:

$$Mix_{bern}(h_1, h_2) = mh_1 + (1 - m)h_2 \tag{5}$$

where m is sampled from a Bernoulli(p) distribution (p can simply be sampled uniformly) and multiplication is element-wise.

3 Dataset

We use 3 datasets to evaluate the representations learned from the above method.

- MNIST: It consists of images of handwritten digits of size 28 * 28.

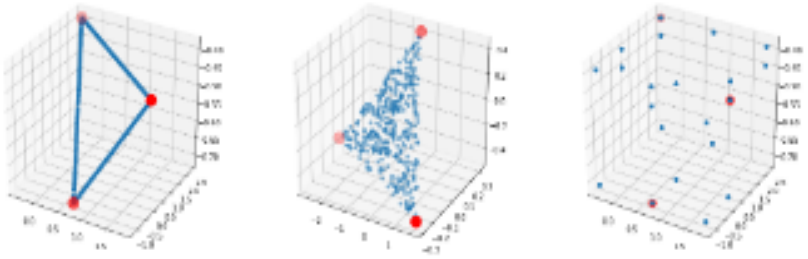

Figure 1: Left: mixup (Equation 4), with interpolated points in blue corresponding to line segmentsbetween the three points shown in red. Middle: triplet mixup (Equation 6). Right: Bernoulli mixup(Equation 5).

- KMNIST : A dataset which focuses on Kuzushiji (cursive Japanese). It is a drop-in replacement for MNIST. It also has images of size 28*28.
- SVHN: This is the street view house number datset. It is similar to MNIST, it consists of images of size 32 * 32.

For all the datasets, normalizing the images was the only preprocessing step that we applied.

4 Experiments and Results

The performance of the proposed concept is evaluated via downstream classification tasks. A linear classifier probe is added to the end of encoder and is trained in unison with the auto-encoder and discriminator but it does not contribute any gradients to the encoder. The accuracy is taken as a measure to quantify the usefulness of the representation learned by the encoder. The datasets employed for classification are MNIST,KMNIST and SVHN. We report the highest accuracy achieved on the validation set.

Hyperparameter values are the same as those suggested by the authors and have not been changed for correct reproduction of the results. We keep k constant and equal to two due to time and performance constraints but we believe the results obtained are enough to validate the reproducibility of the task at hand. We take encoding dimension $d_h = 32$ for all the datasets. The learning rate is varied via a scheduler for smoother training. We start with an intial learning rate of 0.01 and reduced by a factor of 0.1 until 1e-7. Table 1 shows the performance results. For all the experiments we set $\lambda = 10$

Datset	**Mixup Method**	**Accuracy**
MNIST	mixup	94.38
MNIST	bernoulli	94.66
SVHN	mixup	21.58 ± 0.98
SVHN	bernoulli	21.55 ± 0.23
KMNIST	mixup	0.6792
KMNIST	bernoulli	0.6564

Table 1: Test error (%) of Vanilla, Input, Manifold, BC+ Manifold mixup models for different prune percentages on MR dataset

Figure 2: Interpolations between SVHN digits(bottom) and MNIST digits(top). left-most and right-most images are the original images, and images in between are interpolations. First 2 rows in each image show interpolations using bernoulli mixup and the bottom 2 rows show interpolations using manifold mixup. The interpolations are shown for $\alpha = \{0.2, 0.4, 0.5, 0.6, 0.8\}$(left to right)

We also show max variation observed in SVHN(1k) after the accuracy seemed to have stabilized, The accuracy was much more stable in the case of manifold mixup rather than bernoulli and higher as well in case of MNIST.

Figure 2 shows the outputs of both the variants (bernoulli and mixup) for 2 pairs of images. These are generated by combining the latent representatiions of the numbers via manifold mixup and varying α. The results are reasonably close to the images observed in the paper in the case of MNIST and show a reasonably smooth transition in SVHN but the model needs more fine tuning and training.

5 Compute

We used 1 NVIDIA GTX 1070 and 1 NVIDIA GTX 1060 for performing our experiments. Our code is available at `https://github.com/naruarjun/Adversarial-Mixup-Resynthesis`.

6 Conclusion

In this paper we have tried to replicate the experiments in Beckham *et al.* [2019]. They explore different ways of combining the representations learned in auto-encoders through the use of mixing functions. We wrote the entire code from scratch and were able to achieve a performance comparable to that of the original paper.

References

Christopher Beckham, Sina Honari, Alex Lamb, Vikas Verma, Farnoosh Ghadiri, R. Devon Hjelm, and Christopher Joseph Pal. Adversarial mixup resynthesizers. *ArXiv*, abs/1903.02709, 2019.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11, pages 17–37. JMLR.org, 2011.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *ArXiv*, abs/1406.2661, 2014.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states, 2018.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.