LABEL REUSE FOR EFFICIENT SEMI-SUPERVISED LEARNING

Tsung-Hung Hsieh, Jun-Cheng Chen, Chu-Song Chen

Academia Sinica
{andrewhsiehth, pullpull, song}@iis.sinica.edu.tw

ABSTRACT

In this paper, we propose a new learning strategy for semi-supervised deep learning algorithms, called label reuse, aiming to significantly reduce the expensive computational cost of pseudo label generation and the like for each unlabeled training instance since pseudo labels require to be repeatedly evaluated through the whole training process. For label reuse, we first divide the unlabeled training data into several partitions, replicate each partition in several copies, and place them consecutively in the training queue so as to reuse the pseudo labels computed at first time before invalidation. To evaluate the effectiveness of the proposed approach, we conduct extensive experiments on CIFAR-10 [1] and SVHN [2] by applying it upon the recent state-of-the-art semi-supervised deep learning approach, MixMatch [3]. The results demonstrate the proposed approach can not only significantly reduce the cost of pseudo label computation of MixMatch by a large amount but also keep comparable classification performance.

***Index Terms*—** label reuse, semi-supervised learning, MixMatch

1. INTRODUCTION

In the past few years, deep learning-based algorithms have achieved promising results in various computer vision and machine learning applications, including image classification [4, 5], object detection and semantic segmentation [6], image generation [7, 8], etc. However, to learn a robust model through deep learning usually requires a huge amount of labeled training data which is costly to be collected. On the other hand, there are plenty of unlabeled data innately available; thus, how to effectively leverage those unlabeled data for effective training is an important and ongoing research problem. For pure unsupervised learning, there still exists a large performance gap from supervised learning currently. Another promising research direction is semi-supervised learning which makes use of a small amount of labeled data with a large amount of unlabeled data for training, and many researchers have found this setting helps achieve considerable improvement in performance.

In general, semi-supervised learning consists of the following main steps: (1) to train an initial classifier using a limited amount of labeled training data, (2) to evaluate it on unlabeled data followed by choosing the most confident class (or build links between labeled and unlabeled data and to propagate the labels from known instances to unknown ones) (3) to add the high-confidence samples into the training set, and (4) to repeat the same training process until convergence or the maximum number of the allowed iterations reached. The second step is the so-called pseudo label computation which uses the current model to infer the labels of unlabeled samples, and the inferred labels are then used to train the classifier. Recently, following this learning paradigm, there are several semi-supervised deep learning methods proposed which leverage the characteristics of label consistency after simple data augmentation techniques to compute pseudo labels (*e.g.*, the recent state-of-the-art semi-supervised deep learning approach, MixMatch [3], averages the classification outputs as the final pseudo label after applying K different augmentation operations to each unlabeled instance, followed by non-linear transformation-based post-processing.) or similar supervision (*e.g.*, [9, 10, 11] propose consistency loss of the classification outputs of different data augmentations.).

It has been shown that these approaches can effectively train a deep learning model with better or similar performance compared to other fully supervised methods using few labeled samples (*e.g.*, 250 labeled instances for CIFAR-10 [1]) and unlabeled ones. However, these methods require to repeatedly evaluate the pseudo labels for each unlabeled training instance during the whole training process, and this operation will introduce a lot of additional cost and noise into training. To address this problem, we propose a new learning strategy for semi-supervised learning, called label reuse. For label reuse, we first divide the unlabeled training data into several partitions, replicate each partition several times, and place them consecutively in the training queue so as to reuse the pseudo labels computed at first time before invalidation. Besides, we can apply label sparsification to suppress the low-confidence responses in each pseudo label which can save the memory usage for buffering the precomputed pseudo labels and sometimes help mitigate the influence of noisy labels. The overview of the approach is shown in Fig. 3 and will be discussed in detail in Section 3. We conduct extensive experiments on CIFAR-10 [1] and SVHN [2] by applying the proposed approach upon the representative semi-supervised learning algorithm, MixMatch [3]. The results demonstrate the effectiveness of the proposed approach which can not only significantly reduce the cost of pseudo label computation of MixMatch by a large amount but also keep comparable classification performance.

The paper is organized as follows. In Section 2, we briefly review recent relevant works for semi-supervised learning of image classification. In Section 3, we describe the details of the proposed approach leveraging data rearrangement to greatly reduce the number of pseudo label estimation for unlabeled data and to speed up the training process. In Section 4, we conduct a series of experiments to evaluate the effectiveness of the proposed approach. Finally, we conclude the whole paper in Section 5.

2. RELATED WORK

In this section, we briefly review recent relevant deep learning works in the semi-supervised learning setting [12] which can be divided into two main categories, (1) consistency loss-based regularization

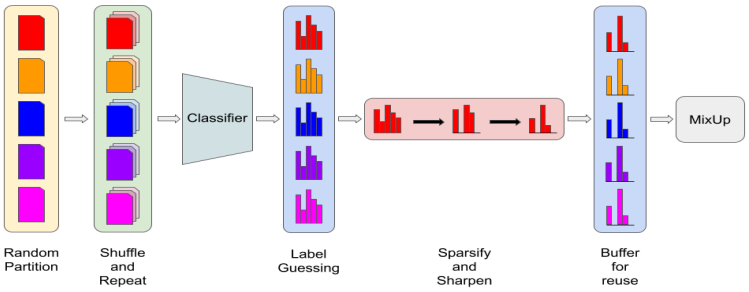

Fig. 1. An overview of the proposed approach where it demonstrates the setting with the number of partitions $p = 5$ and the number of repetitions $r = 3$. In the random partition part, we split unlabeled data into 5 partitions (in different colors). We then replicate each partition in three times, and place them consecutively in the training queue so as to reuse the pseudo labels computed at first time before invalidation. In addition, for each new partition, we perform individual data shuffling. Once the data is prepared, we follow the setting of the original MixMatch algorithm to apply Mixup to mingle labeled and unlabeled data for training. The figure is best viewed in color.

and (2) pseudo label assignment to the unlabeled data.

Consistency loss-based regularization: Consistency loss generally encourages the outputs of two similar images through a classifier should also be similar, and this type of loss can be used as an unsupervised loss for semi-supervised deep learning. For image classification, although various data augmentation operations are usually used to increase the size of training data and change the pixel content of an image a lot, they do not alter the label. Thus, [13, 14, 15, 9] propose to enforce the consistency loss between the outputs of a set of random perturbations or transformations of an unlabeled image. On the other hand, since most of the methods mentioned above use domain-specific data augmentation, instead, Miyato *et al.* [16] propose virtual adversarial training (VAT) to compute an additive perturbation which will maximally change the output class distribution as a better data augmentation. Recently, Xie *et al.* [10] also follows similar paradigm as [16] to further study a rich set of data augmentation for different tasks on computer vision and natural language processing and achieves promising results on the benchmarks of these tasks.

Pseudo Label Assignment: Once the pseudo labels of unlabeled data are attained, they can be used as the ground-truth labels annotated by humans to train the deep networks using commonly used cross entropy loss like [17]. To acquire the pseudo labels, Lee *et al.* [17] propose to firstly apply the currently learned classifier to unlabeled training data followed by selecting the most confident class as the pseudo labels. On the other hand, [18, 19] perform label propagation to produce the pseudo labels for unlabeled data based on the descriptors of the pre-trained deep convolutional neural network (DCNN). Recently, Berthelot *et al.* [3] propose a method called MixMatch which obtains the pseudo label by aggregating K data augmented copies from an unlabeled training instance. Then, Mixup [20] is further used to mingle labeled and unlabeled data for training, and their approach achieves state-of-the-art performance on several benchmarks.

Regardless of the approaches of consistency loss-based regularization or pseudo label assignment, they all require to repeatedly attain certain outputs from the most updated classifier for further processing. Although we mainly use the proposed label reuse strategy to MixMatch, it can also be applied to other approaches with this nature by simple adaptation.

3. PROPOSED METHOD

In this paper, we focus on speeding up the recently proposed state-of-the-art semi-supervised deep learning algorithm, MixMatch, which performs label-preserving data augmentation operations followed by feeding into the classification model to attain pseudo labels and to use them as supervisory signals. However, this process is usually costly and needs to be frequently re-estimated for each training instance. To address this issue, we proposed label reuse strategy which firstly divides the unlabeled training data into several partitions, replicate each partition in several copies, and place them consecutively in the training queue so as to reuse the pseudo labels computed at first time before invalidation to save the cost of pseudo label computation. In the following subsections, we describe each component of the proposed approach in detail.

3.1. Label Reuse

In order to reuse estimated labels, we duplicate the unlabeled training data, and the estimated labels will be shared by all the duplicated data. The details are explained as follows.

Data Preparation: We randomly split the unlabeled data $\mathcal{D}$ into p partitions of equal size.

$$\mathcal{D} = [\mathcal{P}^1, ..., \mathcal{P}^p]. \tag{1}$$

We then generate an r times larger dataset $\tilde{\mathcal{D}}$ by repeating and shuffling each partition $\mathcal{P}^i$.

$$\tilde{\mathcal{D}} = [\mathcal{S}^1, ..., \mathcal{S}^p]. \tag{2}$$

$$\mathcal{S}^i = [\text{shuffle}(\mathcal{P}^i)_1, ..., \text{shuffle}(\mathcal{P}^i)_r]. \tag{3}$$

We apply a random shuffle operation to each repeated partition to avoid the occurrence of the same data pattern. The model is firstly trained on $\mathcal{S}^1$ and then $\mathcal{S}^2$ and so on. After visiting all the data in $\tilde{\mathcal{D}}$ we will create a new $\tilde{\mathcal{D}}$ using different random partitions.

Computational Cost Reduction: In order to reduce the computational cost for forwarding k-augmented data in MixMatch, for each repeated sequence $\mathcal{S}^i$, we only estimate the labels for unlabeled data once during the training of the first repetition $\text{shuffle}(\mathcal{P}^i)_1$. The estimated labels are then kept in a buffer and are reused for the training of the rest of the following shuffled repetitions, $\text{shuffle}(\mathcal{P}^i)_j$, where $j = 2, ..., r$.

Note that during the training of $\mathcal{S}^i$, we will access only one partition rather than the whole of unlabeled data, so even though our method requires some extra memory to buffer the estimated labels, the overhead is not much.

3.2. Label Sparsification

In order to reduce the introduced memory overhead of buffering estimated labels, we can optionally apply label sparsification to sparsify each estimated label by keeping only the t-largest values in it and truncate the rest of values to zeros, and then we perform the sharpening operation on the sparsified label.
First, we obtain the guessed labels the same as done by MixMatch:

$$\bar{q}_b = \frac{1}{K}\sum_{k=1}^{K} \mathrm{p}_{model}(y|\hat{u}_{b,k};\theta), \tag{4}$$

where $\hat{u}_{b,k}$ are the unlabeled training instances for b-th batch and k-th augmentation (*i.e.,* K data augmentations are used in MixMatch), and $\bar{q}_b$ are the corresponding estimated pseudo labels. Secondly, we sparsify the label:

$$\tilde{q}_b = \text{KeepTopK}(\bar{q}_b). \tag{5}$$

Last, we sharpen the sparsified label:

$$q_b = \text{Sharpen}(\tilde{q}_b, T), \tag{6}$$

$$\text{Sharpen}(p, T) = \frac{p_i^{\frac{1}{T}}}{\sum_{j=1}^{L} p_j^{\frac{1}{T}}}. \tag{7}$$

We can then either use this sharpened sparsified label or unsparsified one in the following MixUp stage.

3.3. MixMatch Loss Function

Once the data and pseudo labels are prepared, we follow the setting of the original MixMatch algorithm to apply Mixup to mingle labeled and unlabeled data for training. As shown in the following equations, We follow the loss functions used in MixMatch.

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|}\sum_{x,p\in\mathcal{X}'} \mathrm{H}(p, \mathrm{p}_{\text{model}}(y|x;\theta)). \tag{8}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|}\sum_{u,q\in\tilde{\mathcal{U}}'} \|q - \mathrm{p}_{\text{model}}(y|u;\theta)\|_2^2. \tag{9}$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}}\mathcal{L}_{\mathcal{U}}. \tag{10}$$

where $\mathcal{X}'$ and $\mathcal{U}'$ are obtained as

$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha). \tag{11}$$

And the following is how MixMatch generate $\mathcal{X}', \mathcal{U}'$:

$$\hat{\mathcal{X}} = ((\hat{x}_b, p_b); b \in (1, ..., B)). \tag{12}$$

$$\hat{\mathcal{U}} = ((\hat{u}_{b,k}, q_b); b \in (1, ..., B), k \in (1, ..., K)). \tag{13}$$

$$\mathcal{W} = \text{Shuffle}(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})). \tag{14}$$

$$\mathcal{X}' = (\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, ..., |\hat{\mathcal{X}}|)). \tag{15}$$

$$\mathcal{U}' = (\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, ..., |\hat{\mathcal{U}}|)). \tag{16}$$

where $\hat{x}$ are the augmented labeled data, $\mathcal{X}'$ and $\mathcal{U}'$ are the processed batches produced by MixMatch which mingles both labeled and unlabeled training data, and $\mathcal{L}_{\mathcal{X}}$ and $\mathcal{L}_{\mathcal{U}}$ are the cross entropy loss for $\mathcal{X}'$ and the consistency loss for $\mathcal{U}'$, respectively. The difference here is that we use our shuffled and repeated dataset instead of the original unlabeled dataset. For more details of MixMatch, we refer to interested readers to [3].

4. EXPERIMENTAL RESULTS

In the following subsections, we present the evaluation results and ablation studies of the proposed approach.

4.1. Experimental Settings

We follow the experimental setting in MixMatch with slight modification. The classification model used in this work is the 28-layer Wide Resnet model [21] which has only 1.5 million parameters and we use AdamW as the gradient optimizer. All results are trained with 1024 epochs which results in a total $1,048,576 = 1024 \times 1024$ iterations to make the comparison fair. All hyperparameters are the same as MixMatch suggested. For CIFAR-10, $\lambda_{\mathcal{U}} = 75$ and $\alpha = 0.75$. For SVHN, $\lambda_{\mathcal{U}} = 250$ and $\alpha = 0.75$. The sharpening temperature $T = 0.5$, the number of unlabeled augmentations $K = 2$, weight decay 0.02, learning rate 0.002, batch size 64 are used for all experiments and 5000 training examples are held out for validation. We warm up the model by training with labeled data only for 5 epochs.

4.2. Evaluation Comparison with Different Methods

Here, we compare the training accuracy with other methods and different numbers of labeled data. The results on CIFAR-10 are shown in Table 1 and the results on SVHN are shown in Table 2. Our method does not require the label estimation at each iteration; however, the performance is still comparable to the original MixMatch method.

Accuracy on CIFAR-10		
Method/# of labeled data	250	4000
PseudoLabel [17]	50.02	83.79
MixUp [20]	52.57	86.85
VAT [16]	63.97	88.95
MeanTeacher [9]	52.68	89.64
MixMatch [3]	88.92	93.76
Ours (p=5, r=2)	87.18	93.77

Table 1. Comparison between different methods on CIFAR-10

Accuracy on SVHN		
Method/# of labeled data	250	4000
PseudoLabel [17]	78.84	94.29
MixUp [20]	60.03	92.04
VAT [16]	91.59	95.80
MeanTeacher [9]	93.55	96.61
MixMatch [3]	96.22	97.11
Ours (p=5, r=3)	95.45	96.73

Table 2. Comparison between different methods on SVHN.

4.3. Performance of Different Number of Repetitions

We run a different setting with 250 labeled data where the number of partitions is fixed to 5 and set the number of repetitions to $2, 3, 4$. The following table gives the results.

Since we share the same estimated label between different repetitions, the number of forwarding for label guessing can be reduced according to r. For example, it takes about $5,242,880$ and

Accuracy for different number of repetitions			
Dataset / # of repetitions	2	3	4
CIFAR-10	87.18	86.99	86.12
SVHN	95.36	95.45	95.33

Table 3. Comparison between different number of repetitions.

$3,670,016$ forward passes during the whole training process for $r = 1$ and $r = 4$, respectively, which is a 30% reduction. For our wide Resnet model, this will save about $(5,242,880-3,670,016) \times 215.78$MFlops $= 339,392$GFlops during the training.

One can imagine that, as the number of repetitions increase, the performance should eventually start to deteriorate at some point since we are updating the model with label estimations that are too stale. Therefore, deciding the number of repetitions is a trade-off between computational cost and performance.

4.4. Performance of Different Number of Partitions

In this experiment, we also use only 250 labeled data. The number of repetition r is fixed to 2 and the number of partitions is set to 1, 5, and $\frac{N}{\text{batch size}}$ (699 for CIFAR-10 and 1062 for SVHN), where N is the number of unlabeled data, we refer to this as "max" in the table. The "max" setting can be viewed as we immediately reuse the estimated label for the same batch of data but with different data augmentations. Following we show the accuracy of different numbers of partitions.

Accuracy for different number of partitions			
Dataset / # of partitions	1	5	max
CIFAR-10	86.57	87.18	85.30
SVHN	95.16	95.36	95.01

Table 4. Comparison between different number of partitions.

As the table shows, the number of partitions $p = 5$ gives a better performance than the other two extreme cases. When $p = 1$, it is essentially the original unlabeled dataset. On the other hand, in the "max" setting, it can be viewed as each partition contains only one batch of data, and the estimated labels are consecutively reused r times across the same data with different augmentations.

4.5. Performance of Different Degree of Sparsity

We test the effect of keeping only the top-t values in the guessed label and truncating the rest.

In this experiment, we use the best settings in the previous subsection, that is, for CIFAR-10, we use $p = 5, r = 2$, and for SVHN, we use $p = 5, r = 3$.

Accuracy for different sparsity			
Dataset / Top t	6	8	10
CIFAR-10	86.40	87.43	87.18
SVHN	95.31	95.33	95.45

Table 5. Comparison between different sparsity.

In Table 5, the column "Top-10" gives the result of using the non-sparsified label. By keeping only Top-6 values in the guessed label, the extra memory needed for buffering is reduced by 40 %, however, it does not harm the performance a lot. By applying the sparsification strategy, the resulting labels will be sharper than the non-sparsified ones. As a result, the model has to give a more confident label estimation in order to minimize the loss term. This may explain the better performance on CIFAR-10 by keeping only top-8 values in the guessed label.

4.6. Training Curves

Next, we show the training accuracy curve in our experiments. According to Fig 2 and Fig 3, our training converges slower in the beginning when the repetition r is larger. However, in the end they all converge to similar performance level (*i.e.,* we can find the gap of training accuracies of different training curves are in acceptable ranges in the end.). This demonstrates the effectiveness of the proposed method.

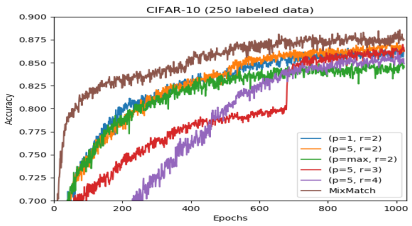

Fig. 2. Training Curve on CIFAR-10.

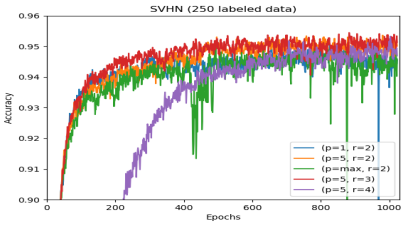

Fig. 3. Trainin Curve on SVHN.

5. CONCLUSION

In this paper, we propose a new learning strategy, label reuse and sparsification, for semi-supervised deep learning algorithms. Through the extensive experiments with recent state-of-the-art MixMatch algorithm, it shows the proposed approach can not only effectively reduce the expensive computational cost of pseudo label generation for each unlabeled training instance and the influence of noisy labels by a large amount but also keep similar performance as MixMatch. On the other hand, the parameters of the size and the number of repeated times for each partition are fixed once they are set at the beginning. However, the initial pseudo labels are much less reliable and much noisier than those at later epochs, we thus plan to make the parameters adaptively adjustable to save more computation as the future work.

6. ACKNOWLEDGEMENT

This work is partially supported by Ministry of Science and Technology, Taiwan, R.O.C. under Grants no. MOST 108-2218-E-001-004-MY2 and MOST 109-2634-F-001-009.

7. REFERENCES

[1] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," .

[2] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[3] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition(CVPR)*, 2016, pp. 770–778.

[5] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2018, pp. 7132–7141.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE international conference on computer vision (ICCV)*, 2017, pp. 2961–2969.

[7] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *International Conference on Learning Representations (ICLR)*, 2019.

[8] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4401–4410.

[9] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Advances in neural information processing systems (NIPS)*, 2017, pp. 1195–1204.

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

[11] W. Shi, Y. Gong, C. Ding, Z. MaXiaoyu Tao, and N. Zheng, "Transductive semi-supervised deep learning using min-max features," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 299–315.

[12] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *MIT Press*, 2006.

[13] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Mutual exclusivity loss for semi-supervised deep learning," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 1908–1912.

[14] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 1163–1171.

[15] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," *International Conference on Learning Representations (ICLR)*, 2017.

[16] T. Miyato, S. Maeda, M. Koyama, and S. Ishii, "Virtual adversarial training: a regularization method for supervised and semi-supervised learning," *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, vol. 41, no. 8, pp. 1979–1993, 2018.

[17] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, 2013, vol. 3, p. 2.

[18] M. Douze, A. Szlam, B. Hariharan, and H. Jégou, "Low-shot learning with large-scale diffusion," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3349–3358.

[19] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5070–5079.

[20] H. Zhang, M. Cisse, Y. N Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *International Conference on Learning Representations (ICLR)*, 2018.

[21] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.