Application of Segmentation based Techniques in Performance Optimization of Deep Neural Networks

Kshitij Bharat Sanghvi
Department of Computer Science
Courant Institute of Mathematical Sciences
NY, USA
kbs391@nyu.edu

***Abstract*—Modern architectures are capable of learning powerful representational spaces. However, due to the model capacity required to capture such representations, they are often susceptible to overfitting and therefore require regularization to generalize well. In this project, we show that the cutout regularisation in junction with batch augmentation, complements our approach of augmenting database by segmenting foreground which can be used to improve the robustness and overall performance of convolutional neural networks. We evaluate this method by applying it to Residual Networks on the CIFAR-10 data-set yielding improved generalization performance.**

I. Introduction

The last decade witnessed a significant boost in deep learning especially in the field of supervised image classification. Pivoting work by Alex Krizhevsky [1] ignited an entire sector of image classification challenges. This race led to the introduction of several different architectures such as VGG [3] and Inception [4] models. In the field of computer vision, challenging vision tasks such as object recognition [8], semantic segmentation [11], image captioning [16], and human pose estimation [15] are now all feasible with current hardware. A large portion of the success can be attributed to the use of convolutional neural networks (CNNs) [9], which are capable of learning complex hierarchical feature representations of images. Modern networks such as VGG [3] commonly contain on the order of tens to hundreds of millions of learned parameters which provide the necessary representational power for such tasks. But the straight side effect is the possibility of overfitting and consequently poor generalization.

To mitigate overfitting different regularization techniques are available such as data augmentation. Data augmentation is prevalent because of its effectiveness. Image transformations such as rotation or scaling can be applied to create new training data which can be used to improve model robustness and increase accuracy [9]. Alternatively, regularization by inducing noise is another common approach. Techniques such as dropout have shown a great performance boost in MNIST classification [16]. In this project, we have used foreground segmentation to further augment our database. This technique has the same advantage as before but it also serves as a complementary approach to cutout. Where cutout randomly masks certain portions of the image, foreground segmentations roughly point to the target feature based on the assumption that the feature is present in the foreground which is generally the case and is fairly segmented by our technique. Our approach is novel and tries to show the network what it needs to learn. By using it in conjunction with cutout, it is like providing the missing piece that completes the jigsaw puzzle. So, next time if we have to extract that piece the neural network has an improved chance. We back the stated claim with empirical evidence as it results in improved generalization performance.

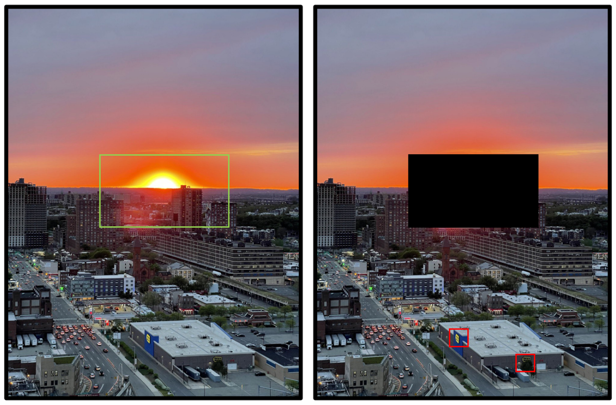
Fig. 1. Cutout Regularisation

II. Related Work

Coming to cutout regularisation, it is a simple regularization technique that involves removing contiguous portions of input images, i.e partially occluded versions of existing samples are appended to the dataset with the same target label. This technique can be interpreted as an extension of dropout. As we are masking a certain portion of the image it is equivalent to dropping the input activations at the input layer itself. There are differences that in drop out it is probabilistic and not all layers experience the same occlusion. Here, no layer has any connection to the portion of the image that has been cut out.

Our work stems from these two techniques that are proven to be effective regularization technique. In the early days, while training LeNet5 [17] for optical character recognition, LeCun et al. applied various affine transformations, including horizontal and vertical translation, scaling, squeezing, and horizontal shearing to improve their model's accuracy and robustness. Thus, data augmentation does exactly what it says. It augments the data by applying transformations to images in a way that the target label still holds for the semantic feature present in the image. For instance, if you look at the following figure, let us say that the image is a pencil. Now, if we apply a rotation transformation the resulting image is still a pencil. That is the target label still holds the semantic meaning. Such transformations may not be effective for symmetrical objects. In [18], Bengio et al. demonstrate that deep architectures benefit much more from data augmentation than shallow architectures. So, it makes more sense to work with our Residual Networks which can be deeper as they have an identity mapping and thus, the problem of vanishing gradient is alleviated. People have also used neural networks to implement the concept of batch augmentation to prevent its identical mode of operation with all datasets. Lemley et al. tackle the issue of data augmentation with a learned end-to-end approach called Smart Augmentation [19] instead of relying on hard-coded transformations. In this method, a neural network is trained to intelligently combine existing samples to generate additional data that is useful for the training process. We use the simplest form of batch augmentation as stated earlier.

Deeper neural networks are more difficult to train. He et al. presented a residual learning framework to ease the training of networks that are substantially deeper than those used before. We will refer them as ResNets [20]. They reformulated the layers as learning residual functions concerning the layer inputs, instead of learning unreferenced functions. They provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. The depth of representations is of central importance for many visual recognition tasks.

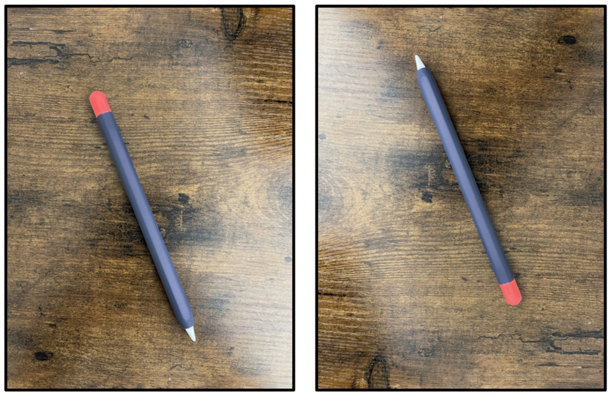

Fig. 2. Batch Augmentation - Rotation

III. Motivation

In the context of learning, it is desired that the model should not only provide good training accuracy but also good generalization performance. Over-fitting should be avoided. Even if the training times are elevated, a boost in accuracy and generalization is deemed as a feasible approach in our outlook. Train once, infer always. Of course, drift detection may involve retraining but we are concerned here solely about the performance. This gives us the freedom to work with augmented data sets and smaller learning rates.

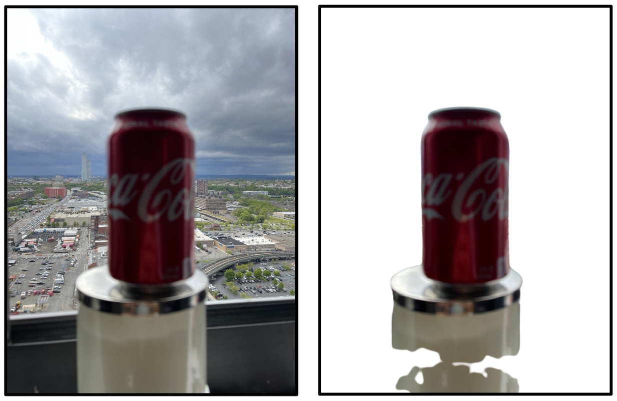

Fig. 3. Foreground Extraction

We think that cutout regularization is kind of a regularization technique that, in a way, is complementary to our approach. In the paper, How transferable are features in deep neural networks? [6] it is stated that a deep learning model performs better if it has already seen the data before. We build on this intuition and it also serves as our motivation that we can incorporate foreground segmentation to boost our performance.

IV. Implementation

For our implementation, we have used TensorFlow and PyTorch using a combination of underlying hardware. We worked with several segmentation algorithms. Choice of segmentation algorithm was grabcut, using the OpenCV implementation. Our choice of TensorFlow backend was motivated by the simplicity of use and availability of compatible libraries with the underlying hardware. However, as we were experimenting we were receiving extended training times. To mitigate the issue we looked at a few pruning algorithms. The drawback of TensorFlow is the unavailability of a straightforward method to extract the activation values. So we switched our framework of choice to PyTorch. To find a suitable solution to switch frameworks, we came across ONNX, Open Neural Network Exchange, that offers APIs for model inter-convertibility easily and intuitively. Another problem that we faced is the unavailability of appropriate GPU. Ellesmere Raedon has not yet received full support from CUDA. Hence, we had to use docker as our local GPU was Radeon RX580. After extended effort, we migrated to cloud technologies. We deployed our model on GCP and experimented with various Nvidia GPUs such as P100, V100, T4, and K80. We also tried TPU but the performance was not up to the expectation.

For our choice of optimization algorithm, we chose RMSProp. Our base architecture is ResNet 44. ResNet helps us in several ways and eliminates everyday problems like initialization, vanishing gradients. The reason why sensitivity

TABLE I
LEARNING RATE SCHEDULER

Drop Counts	Learning Rate	Epoch
1	10^{-3}	1-50
2	10^{-4}	51-65
3	10^{-5}	66-75
4	10^{-6}	76-85
5	$5e^{-7}$	86-100

to initialization is reduced is that ResNets have an identity function, so when the gradients are back-propagated, they do not vanish easily for the initial layers like the others making it somewhat robust to initialization. We have found the optimum batch size to be 64 images. The M value for the cutout that gave us the optimum result is 4. We have opted for a dynamic learning rate scheduler that drops the learning rate value after a few epochs as shown in the table. Empirically, fine-tuning gave us inferior results. Code here. We had to train the model from scratch and run a grid search for the hyper-parameters, which was time-consuming and computationally expensive to run. The performance details are delineated in the following section.

V. Experimental Evaluation

We have used training and validation accuracies and losses for our formal basis of comparison. We have compared results for plain cutout regularisation, cutout regularisation coupled with batch augmentation, and finally the trio - cutout, batch augmentation, and foreground extraction. As we can see from the figure below. The best training loss is obtained for the plain cutout. This in sync with the training accuracy. Similarly, we can see that the next best training performance is cutout and batch augmentation followed by our trio. They are also correctly reflected in the training accuracy. Training accuracy is not a good measure of performance as the desired quality is good generalization. So, we bring validation performance into the picture. We can see that our trio strategy has the minimum loss out of all and this correctly translated to the boost in accuracy performance. We have received close to 90% accuracy which is the highest and serves as empirical evidence to our above claim. The sudden drop at epoch 50 is due to reduced learning rate.

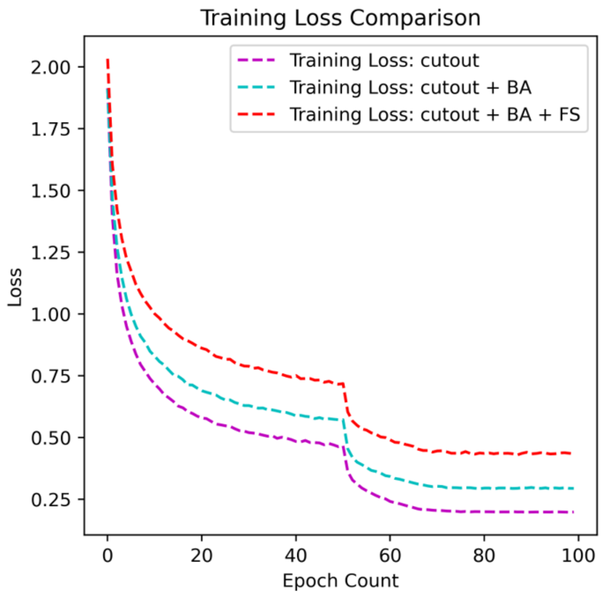

Fig. 4. Training Loss Comparison.

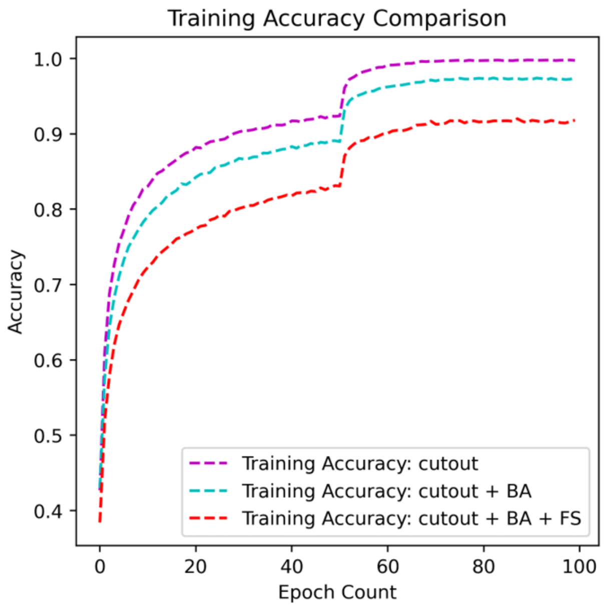

Fig. 5. Training Accuracy Comparison.

VI. Pipeline Deployment

With the introduction of the ONNX[8] framework, a single model for a particular hardware configuration can be morphed into a different framework and hardware device. We now propose a pipeline architecture that can utilize the aforementioned approach and boost performance. A pipeline receives the model and the dataset as input. It first converts the input model into its preferred framework using the ONNX API, in our case TensorFlow. It then runs the steps outlined in the above section. The result is the trained model. Finally, the model can be transformed into .onnx format and deployed at the target hardware where the ONNX Run-time environment [ORT] can be used for inferencing which automatically optimizes depending on the underlying platform. The pipeline is summarised in Figure and can be realized with a single-click deployment.

The option to convert to a specific model still stands. It is not necessary to deploy the same on the ORT server.

VII. Conclusion and Future work

In this paper, we presented a novel optimisation algorithm that improves performance and integrates well with the existing approaches of regularisation. We also showed the

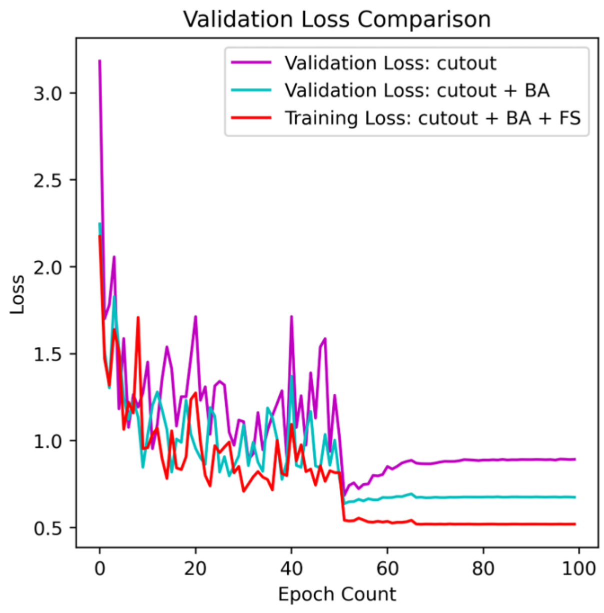

Fig. 6. Validation Loss Comparison.

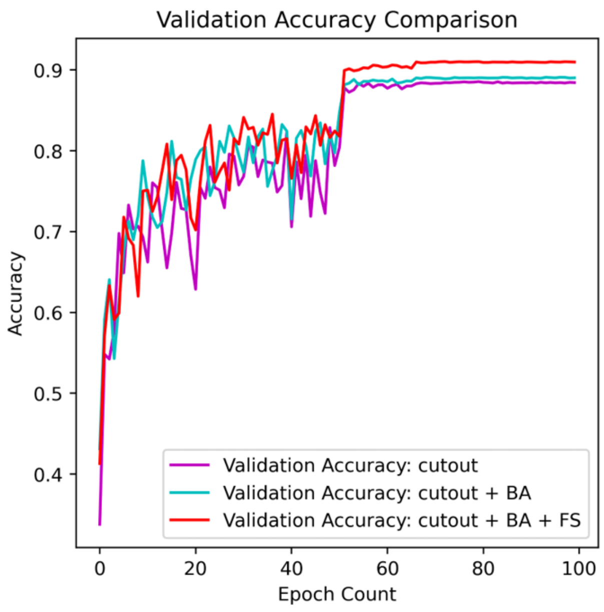

Fig. 7. Validation Accuracy Comparison.

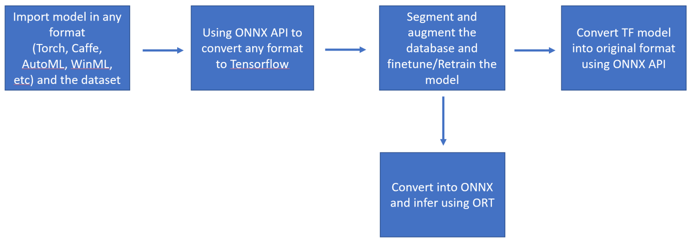
Import model in any format (Torch, Caffe, AutoML, WinML, etc) and the dataset
Using ONNX API to convert any format to Tensorflow
Segment and augment the database and finetune/Retrain the model
Convert TF model into original format using ONNX API
Convert into ONNX and infer using ORT

Fig. 8. Proposed Pipeline.

performance boost. Future work involves testing on larger and more powerful data-sets such as ImageNet[2]. Also, we plan to make the model distributed in nature. Segmenting images of that order is not an easy task. We wish to work on the parameter-server model, distributed training, to see if that helps and find out how the a-sync and sync models trade performance with time.

REFERENCES

[1] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In NIPS, pp. 1106–1114, 2012.
[2] ”ImageNet”, Image-net.org, 2020. [Online]. Available: http://www.image-net.org/. [Accessed: 12- Nov- 2020].
[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.
[4] Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2015.7298594
[5] Bishwaranjan Bhattacharjee, John R Kender, Matthew Hill, Parijat Dube, Siyu Huo, Michael R Glass, Brian Belgodere, Sharath Pankanti, Noel Codella, and Patrick Watson. P2l: Predicting transfer learning for images and semantic relations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 760–761, 2020.
[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In NIPS, 2014.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pages 1097–1105, 2012.
[8] Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. The International Conference on Learning Representations, 2019b.
[9] Md Zahangir Alom, Theodore Josue, Md Nayim Rahman, Will Mitchell, Chris Yakopcic, and Tarek M Taha. 2018. Deep Versus Wide Convolutional Neural Networks for Object Recognition on Neuromorphic System. arXiv preprint arXiv:1802.02608 (2018).
[10] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: bandit-based configuration evaluation for hyperparameter optimization. In International Conference on Learning Representations, 2017.
[11] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, pages 3431– 3440, 2015
[12] ”ONNX — Home”, Onnx.ai, 2020. [Online]. Available: https://onnx.ai/. [Accessed: 14- Dec- 2020].
[13] Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12:2121–2159, July 2011. ISSN 1532-4435.
[14] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In CVPR, pages 1653–1660, 2014.
[15] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In CVPR, pages 3156–3164, 2015.
[16] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012
[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradientbased learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.
[18] Y. Bengio, A. Bergeron, N. Boulanger-Lewandowski, T. Breuel, Y. Chherawala, et al. Deep learners benefit more from out-of-distribution examples. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pages 164–172, 2011.
[19] J. Lemley, S. Bazrafkan, and P. Corcoran. Smart augmentation-learning an optimal data augmentation strategy. IEEE Access, 2017.
[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of CVPR, pages 770–778, 2016. arxiv.org/abs/1512.03385