Exploring Complex Regularization Techniques on Image Classification and Sequence Labelling

Abhinav Gupta Akshita Kapur Hrishikesh Thakur Shreyas Malewar

Abstract

As the scale of modern real-world problems increases, Machine Learning algorithms struggle with overfitting and poor generalization due to a lack of useful data. In this project, we have applied various regularization techniques to a CNN model to perform image classification. Then, we compare and analyze the effect of different regularizations on improving the model's performance. The dataset we have used is Tiny-ImageNet, which has 100,000 images split into 500 classes. The regularization techniques we selected are L2 Regularization, Dropout, Data Augmentation, and Ensemble Regularization (Dropout + L2).

1. Introduction

Machine Learning models often tend to start overfitting to training data during the learning phase of the cycle. Therefore, it is important to keep a constant check, and the best method to do that is through adding regularization. The motivation for this project comes from the lectures of CSCI567.

Through the coursework, we got an opportunity to learn and get hands-on experience with L2 Regularization, a simple method that alters the objective function of any machine learning model to effectively enhance the performance of test data. We were also introduced to several other regularization methods like L1, L0, Data Augmentation, etc. In this project, we have extended what we learned from our assignments to draw a comparative analysis of regularization techniques on a convolutional neural network designed to solve the tiny imagenet challenge.

While it's often believed that copious amounts of data are generated in the modern age, a more analytical perspective reveals that not all available data is useful. For instance, when data acquisition is costly in a tumor segmentation problem, using a complex, high-performing model might lead to overfitting, ultimately reducing its effectiveness. This further piqued our interest in exploring different regularization techniques that help prevent overfitting and effectively utilize all available data.

Therefore, in this project, we aim to understand how the different regularization methods work. We decided to fix the dataset, task, and model to draw a fair and accurate comparative analysis. By only changing the regularization in the algorithm, we can ensure that the change in performance can only be accredited to the change in regularization.

The research article "Empirical Analysis of a Fine-Tuned Deep Convolutional Model in Classifying and Detecting Malaria Parasites from Blood Smears" (Montalbo & Alon, 2021) delves into the application of deep learning for automated malaria parasite detection in blood microscopy images. The authors leverage transfer learning, a technique where a pre-trained deep convolutional neural network (CNN) model, specifically EfficientNetB0 in this case, is fine-tuned for the specific task of malaria parasite classification. This approach is noteworthy for achieving accurate parasite identification with minimal preprocessing steps required for the blood smear images. While this research directly addresses malaria diagnosis, it aligns with broader efforts in image classification that explore complex regularization techniques to enhance the performance and generalizability of deep learning models.

The study titled "Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks(Claudio Filipi Gonçalves dos Santos, 2018)" by Santos and Papa (2022) offers a critical examination of techniques used to combat overfitting in convolutional neural networks (CNNs). Given the susceptibility of complex CNN architectures to overfitting, where models perform well on training data but struggle with unseen data, this survey is particularly valuable. It explores various regularization methods developed in recent years, categorized as data augmentation (manipulating training data), internal changes (modifying the network structure), and label alterations (introducing noise or uncertainty in labels). By analyzing these techniques, the authors provide researchers with a comprehensive toolkit to enhance the generalizability of their CNN models.

Additionally, we also reviewed (Jongga Lee, 2024) work on 'Comparing the effectiveness of regularization methods on text classification: Simple and complex model in data shortage situation'(Jongga Lee, 2024) that helped us under-

stand that regularization also works on different forms of user inputs like text as well, opening a new line of study focus.

2. Methods

2.1. Dataset

The dataset used is *Tiny ImageNet*. A dataset of 100,000+ images split into 200 classes. Each image is downsized to 64×64 colored images. Each class has 500 training images, 50 validation images, and 50 test images. This dataset is a subset of Stanford's Imagenet challenge. The starter code for efficient net B0 is based on images of 224X224. Our challenge was to scale our 64X64 images to fit the pre-existing architecture. We also built a custom convolutional neural network to classify images.

2.2. Regularization

2.2.1. L2

L2 regularization modifies the categorical cross-entropy loss function by adding a term that penalizes the squared magnitude of the weights in the network.

2.2.2. Data Augmentation

Data augmentation introduces variations and distortions to the input samples by applying diverse transformations to the training data. This helps the model learn to be invariant to certain transformations and increases its ability to generalize to unseen data.

2.2.3. Dropout

Dropout is a regularization technique commonly used in neural networks, including Convolutional Neural Networks (CNNs), to prevent overfitting and improve the model's generalization performance. It works by randomly "dropping out" (i.e., deactivating) a fraction of neurons in the network during training.

2.2.4. Ensemble Regularization

It is formed after infusing multiple regularization techniques into one. An ensemble is designed to harness the powers of different well-performing regularization techniques.

2.3. Effecientnet B0

EfficientNet B0 is a lightweight convolutional neural network architecture designed for efficient image recognition. It tackles the challenge of balancing accuracy and computational cost through a unique approach called compound scaling. Using a specific formula, this method scales the network's width (number of channels), depth (number of layers), and input image resolution. As the base for its building blocks, EfficientNet B0 utilizes Mobile Inverted Bottleneck (MBConv) layers, which are known for their efficiency. These layers are further enhanced with Squeeze-and-Excitation (SE) optimization, which focuses the model on informative features within the image data.

Table 1. Final Results Table Compiling Loss at 25th Epoch

Regularization	Training Loss	Val Loss
No Regularization	1.556	2.347
L2 Regularization	1.654	2.1456
Data Augmentation	1.4923	1.9847
Dropout	2.033	2.229
Ensemble	1.72	2.295

2.4. Custom CNN

2.4.1. Convolutional Layers

Convolutional Layers: Feature extraction using convolutions with filter sizes of 11x11, 5x5, and 3x3, comprising 64, 192, 384, 384, and 256 filters respectively, followed by ReLU activation.

2.4.2. Normalization Layers

Incorporating local response normalization after specific convolutional layers, adjusting activations within local neighborhoods, with parameters depth radius of 5, bias of 2.0, alpha of 1e-4, and beta of 0.75.

2.4.3. Pooling Layers

Employing max-pooling with pool sizes of 3x3 and strides of 2 for downsampling, reducing spatial dimensions while maintaining crucial features.

2.4.4. Flattening

Transforming convolutional layer outputs into one-dimensional vectors for seamless integration into fully connected layers.

2.4.5. Fully Connected Layers

Dense layers with 1024 neurons each, activated by ReLU, facilitating complex mappings between flattened features and output classes, culminating in a softmax activation layer for classification.

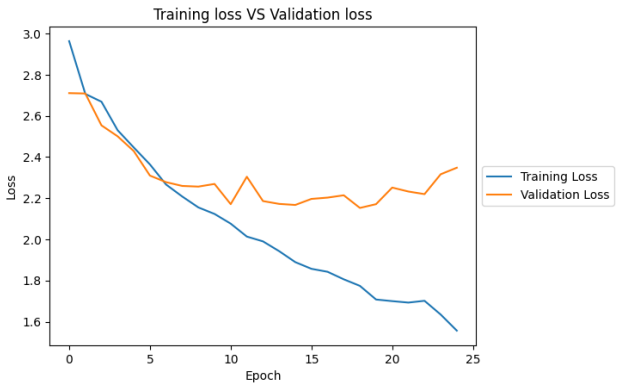

Figure 1. Custom CNN model loss without regularization

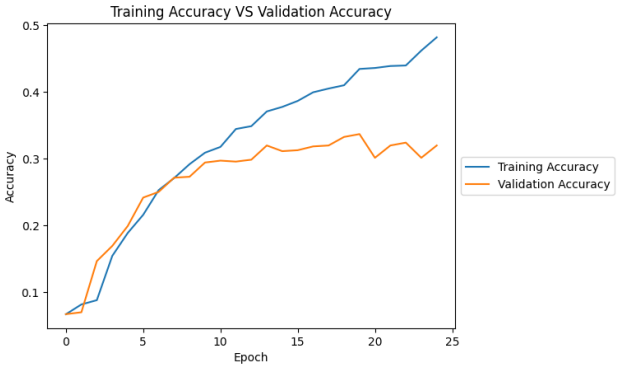

Figure 2. Custom CNN model accuracy without regularization

3. Results

3.1. Custom CNN model without regularization

We can clearly observe a huge generalization gap, indicating overfitting. After the 10th epoch (Figure 1), there is a sudden decrease in training loss and the similar trend is not seen for validation resulting in poor generalization (Figure 2).

3.2. L2 Regularization

After applying L2 regularization, we observed a decrease in the generalization gap for both the loss (Figure 3) and accuracy (Figure 4) curves. Although the training loss continues to decrease and the training accuracy continues to increase, there is a clear improvement in the validation loss and accuracy. With these results, we can conclude that L2 helps reduce the gap and prevent overfitting.

3.3. Dropout Regularization

We applied a dropout rate of 0.25 to both linear layers, which resulted in a significantly reduced generalization gap, as

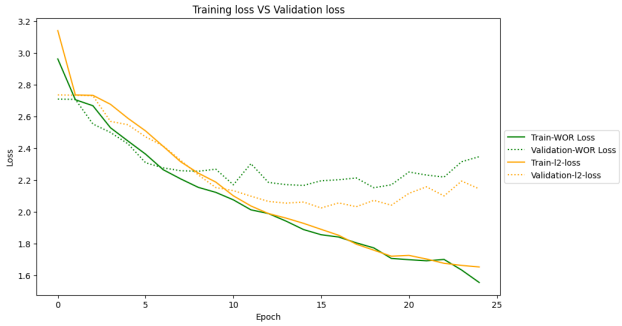

Figure 3. Custom CNN model loss with l2 regularization

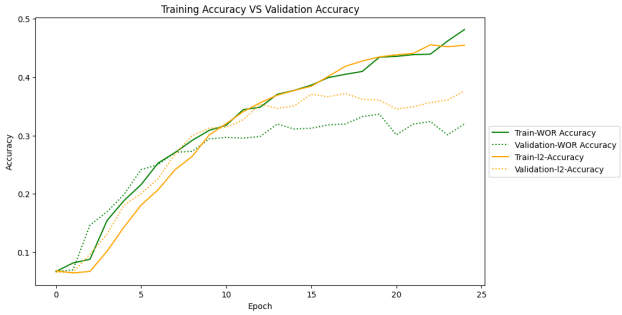

Figure 4. Custom CNN model accuracy with l2 regularization

shown in the graph above. The loss (Figure 5) and accuracy (Figure 6) curves exhibit minimal gaps. Although there is a slight increase in validation loss, it is not substantial enough to negatively affect the model's performance. We considered choosing a lower dropout value due to the rising validation loss, but given the small gap and the insignificant increase in loss, we decided to accept this as our final result.

3.4. Ensemble Regularization

After applying ensemble regularization—specifically, L2 regularization with lambda set to 0.00015 combined with a dropout rate of 0.2—we observed an improvement in the generalization gap (Figure 8). This approach decreased the gap and prevented overfitting more effectively than using dropout alone. Previously, employing just dropout helped reduce the generalization gap but increased validation loss (Figure 7), which we did not experience with the combined approach.

3.5. Data Augmentation

We applied five types of augmentations—left-right and up-down flips, brightness adjustment with a max_delta of 0.4, contrast adjustment with a range of 0.8 to 1.2, and saturation adjustment with a range of 0.7 to 1.3. By implementing

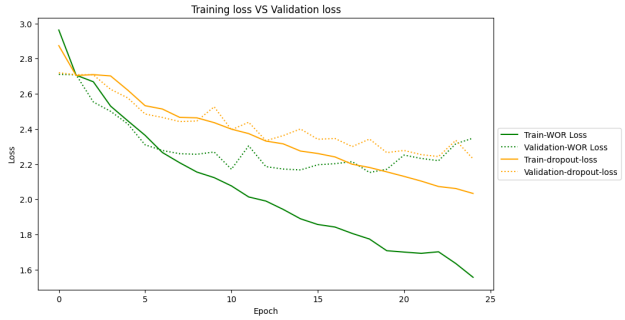

Figure 5. Custom CNN model loss with dropout regularization

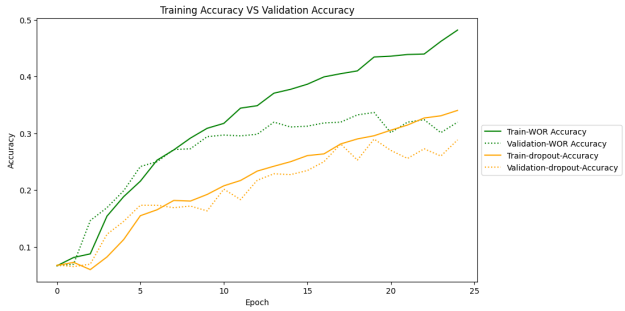

Figure 6. Custom CNN model accuracy with dropout regularization

these augmentations, we effectively doubled our dataset size. The results clearly show a reduction in the generalization gap (Figure 10) and a significant decrease in validation loss (Figure 9) across epochs, which helps prevent overfitting.

The table shows a decrease in validation loss across almost all regularization techniques, with varying degrees. However, the ensemble approach notably increases the training loss (not significantly observed with every technique), narrowing the generalization gap. While dropout contributes to this effect as well, the overall trend for dropout indicates an increase in validation loss from a very early epoch.

3.6. Conclusion

shows a decrease in validation loss across almost all regularization techniques, with varying degrees. However, the ensemble approach notably increases the training loss (not significantly observed with every technique), narrowing the generalization gap. While dropout contributes to this effect as well, the overall trend for dropout indicates an increase in validation loss from a very early epoch.

Figure 7. Custom CNN model loss with ensemble regularization

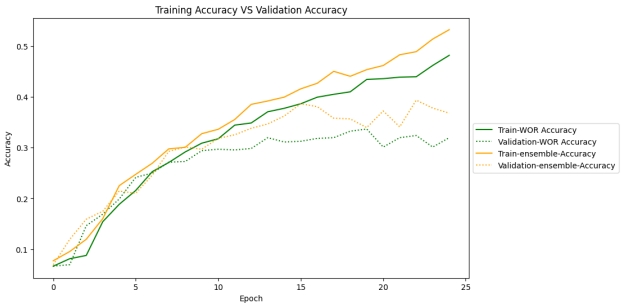

Figure 8. Custom CNN model accuracy with ensemble regularization

4. Process

4.1. Changed Regularization Method

Initially, we planned to use L2 Regularization + Data Augmentation as our ensemble regularization techniques. However, upon further review of the literature (Cai et al., 2022), we realized that this combination was counterproductive. Data augmentation introduces variability to the training data, complicating the training landscape, while L2 regularization simplifies model weights to facilitate easier training. In contrast, dropout, which randomly deactivated nodes during training, enhances feature exploration and diversification, and L2 stabilization controls the size of weights, proving synergistic. Moreover, data augmentation did not yield as good results as the other regularization techniques, leading us to opt for a combination of L2 + MaxDropout instead.

4.2. Pivot in CNN Model

We began our experiment using EfficientNet B0 as our base model for image classification but noticed minimal improvements in model performance after applying regularization. Upon further analysis, we realized that since EfficientNet B0 is a pre-tuned model with built-in regularizations, additional regularization was not particularly beneficial. This led

Figure 9. Custom CNN model loss with data augmentation

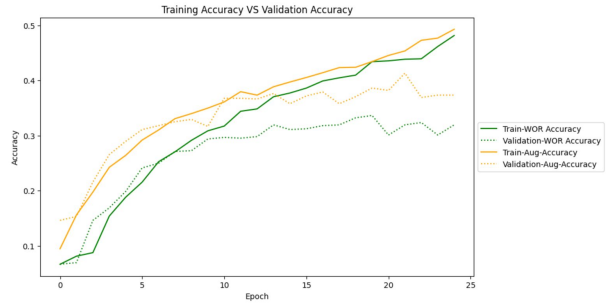

Figure 10. Custom CNN model accuracy with data augmentation

us to switch to our custom CNN model, which we applied regularization.

4.3. Reduced Dataset Scale

We intentionally limited our dataset to only 25 classes to mirror real-life scenarios where data scarcity can lead to overfitting. This smaller dataset allowed us to evaluate whether our regularization strategies effectively reduced overfitting and minimized the generalization gap.

4.4. Dropped Seq2Seq Labeling Task

Initially, we had planned to also perform a second task on a different type of dataset. The task was Seq2Seq Labeling (POS Tagging) on NLTK Corpus dataset using a CNN model. As per our schedule, we decided to work on it after the midterm presentation, however, the first task of image classification was a bigger undertaking than we had anticipated. We had to continue working on improving the implementation of the regularization techniques on the model, improving the model's performance, and hyper-parameter tuning. This led to very little time being available for the second task. We still tried and were able to develop an algorithm that performs POS Tagging of NLTK Corpus using a Vanilla RNN. The model runs very well, with a test accuracy of 95% . However, due to limited time, we were unable to implement the regularization techniques and analyze its effect on the model's performance. Here is the link to the code file and the necessary steps to compile the results.

From the table, we can observe a decrease in validation loss across almost all regularization techniques, with varying degrees. However, the ensemble approach notably increases the training loss (not significantly observed with every technique), which aids in narrowing the generalization gap. While dropout contributes to this effect as well, the overall trend for dropout indicates an increase in validation loss from a very early epoch.

5. Contributions

5.1. Final Code Implementation

We have discussed the bulk of our methodology, results, and analysis in the report's 'Methods' and 'Results' sections. This section will discuss understanding and re-compiling the code to get the results. The final code implementation can be found in this GitHub Repository. The code has sufficient comments explaining the step-by-step process to generate the desired results.

5.2. Packages and Libraries Used

- NumPy - It supports multidimensional arrays and a collection of mathematical functions to operate on these arrays efficiently. NumPy is used extensively for manipulation, preprocessing, and handling image data in arrays.

- Matplotlib - It provides various plotting functions to visualize data in 2D and 3D. Matplotlib is used for data visualization tasks such as plotting images, displaying model performance metrics (e.g., accuracy, loss), and visualizing training progress.

- Pandas - It provides data structures like Data-Frame and Series and functions to manipulate and analyze tabular data efficiently. Pandas are used for tasks such as loading and preprocessing datasets, exploring data distributions, and organizing data into appropriate formats for training and evaluation.

- Sci-kit learn - Scikit-learn is a machine learning library that provides simple and efficient data mining and analysis tools. It includes a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and more. Scikit-learn is used for tasks such as splitting datasets into training and testing sets, applying data preprocessing techniques (e.g., normalization, scaling), and evaluating model performance using metrics like accuracy, precision, recall, etc.

- Keras - It provides a user-friendly interface for building, training, and deploying deep learning models with minimal code. Keras is used to build and train the CNN model. It simplifies defining neural network architectures, configuring model parameters, compiling the model with loss functions and optimizers, and training the model on the dataset.
- TensorFlow - TensorFlow serves as the backend for Keras and provides low-level APIs for implementing custom neural network architectures, optimizing model performance, and deploying models in production environments.

References

Cai, Z., Xie, X., Xie, M., Moshayedi, A. J., and Noori Skandari, M. H. A hybrid improved neural networks algorithm based on l2 and dropout regularization. *Mathematical Problems in Engineering*, 2022. doi: 10.1155/2022/8220453.

Claudio Filipi Gonçalves dos Santos, J. P. P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *Mathematical Problems in Engineering*, 2018. doi: 10.1145/3510413.

Jongga Lee, Jaeseung Yim, S. P. C. L. Comparing effectiveness of regularization methods on text classification: Simple and complex model in data shortage situation. *Computation and Language*, 2024.

Montalbo, F. J. and Alon, A. Empirical analysis of a fine-tuned deep convolutional model in classifying and detecting malaria parasites from blood smears. *KSII Transactions on Internet and Information Systems*, 3, 2021. doi: 10.3837/tiis.2021.01.009.

6. Appendix

1. GitHub Repository - Link