Understanding deep learning requires rethinking generalization: randomization tests

1st Jhon Jairo Silva Zabala
Ingeniería de Sistemas y Computación
Facultad de ingeniería
Bogotá, Colombia
jhsilvaz@unal.edu.co

***Abstract*—This paper explores the generalization capabilities of deep neural networks by employing a series of randomization tests on the CIFAR-10 dataset. We introduce varying levels of noise into the dataset through techniques such as label corruption, pixel shuffling, and the addition of Gaussian noise. Using a smaller version of the Inception neural network, we investigate how these perturbations affect the model's ability to learn and generalize. Our findings reveal that despite high levels of noise, the model can achieve zero training error, indicating its capacity to memorize even random data. However, as expected, the test error increases, demonstrating that generalization is hindered when the data contains significant noise. These results challenge traditional theories of generalization, suggesting that modern deep networks can fit noise while still maintaining some level of signal extraction. This study provides insights into the limits of neural network generalization and highlights the need for new theoretical frameworks to better understand their performance.**

***Index Terms*—Generalization, effective capacity, overfitting.**

I. The randomization test

Looking to show how generalization it is still a problem that has no theoretical basis and how traditional approaches fails to explain why large neural networks generalize well in practice, the team execute a randomization test usign the CIFAR 10 dataset. In the traditional scenario with neural network people work in two steps to build models, first of all is executed an optimization step where using a portion of data called training data if fitting a model, next the model capacity is evaluated using a test data and measure how well it perfoms on newly generated data.

For the experiment, 4 different neural network model architectures were used to create different copies of the original data, introducing some noise to theses. The dirty dataset was created using diferent strategies to introduce some random noise on these, as a result was obtained 6 datasets, there are:

- **True labels:** this dataset contains the original images without any modification.
- **Partially Corrupted labels:** It is a dataset with corrupted labels, as is known CIFAR10 is a dataset of images with label, the labels are assigned random based in a corruption probability, hera was used 20%, 40%, 60% and 80% as corrupted probabilities.
- **Random labels:** it is the dataset when the corruption probability is 100
- **Shuffled pixels:** A fixed random permutation is applied to the pixels of each image, maintaining consistent distortion across all images while disrupting spatial structure, challenging the model's ability to generalize.
- **Random pixels:** Each image undergoes a unique random permutation of pixels, completely scrambling spatial order and increasing unpredictability, making pattern recognition much more difficult.
- **Gaussian Noise:** Images are replaced with synthetic ones composed entirely of Gaussian noise, testing the model's ability to discern patterns within random data that follow the statistical color distribution of CIFAR-10.

The idea behind, is that the randomization breaks any relation between the data, as will be shown later that randomization does not allow to build an eficcient model well with the test error obviously is not 0 because of any relation exists between train and test data. But with training error something different happens and is here where is possible to impresive from the results.

II. Datasets noise

The first step to execute the experiment was to create the datasets with noise using different techniques in each one of them, the following describes for each type of noise how the data obtained from the CIFAR 10 database is fouled.

- **Partially Corrupted Labels:** The idea behind this type of dataset is based on creating for each of the images a random label according to a percentage of randomness. The CIFAR 10 Corrupted class extends the CIFAR10 dataset from torchvision by introducing a mechanism to corrupt the labels of the dataset with a specified probability. Upon initialization, the user provides a corrupt prob parameter that dictates the likelihood of each label being altered. If this probability is greater than zero, the class invokes the corrupt labels method, which randomly changes a subset of the labels to a different class, effectively simulating label noise. This is achieved by generating a mask that identifies which labels to corrupt, and then replacing those labels with randomly selected ones from the set of possible classes.
  This class is particularly useful for evaluating the robustness of machine learning models to label noise. By varying the corrupt prob, one can simulate different levels

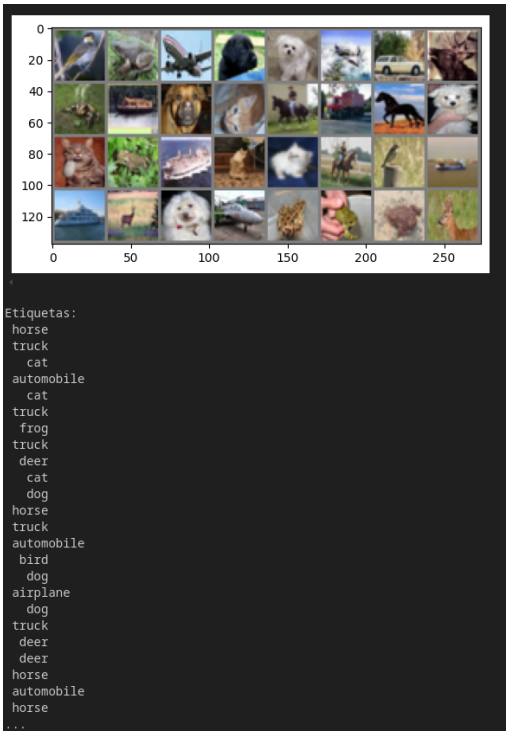

Fig. 1. Enter Caption

of noise in the CIFAR-10 dataset and study how well models can learn and generalize under these conditions. For instance, with a corrupt prob of 0.8, 80% of the dataset's labels will be randomly altered, as is showed in the "Fig. 1", as the reader can see the labels for the images starting from the top left corner are incorrect.

- **Shuffled pixels:** It applies a fixed random permutation to the pixels of each image within a batch. This means that the spatial structure of the images is consistently distorted across the entire batch, but the same permutation is used for all images. This type of perturbation disrupts the normal arrangement of pixels while maintaining a consistent pattern of distortion, challenging the model to learn despite the loss of spatial coherence.
- **Random pixels:** The random pixels option introduces a different form of distortion by applying a randomly generated permutation to the pixels within each batch. Unlike shuffled pixels, where the permutation is fixed, random pixels generates a new random permutation for each batch, leading to more unpredictable and varied distortions. This forces the model to cope with a wider range of spatial disruptions, testing its ability to generalize under more chaotic conditions where the pixel order varies unpredictably.
- **Gaussian noise:** With gaussian noise enabled, the original images are replaced with synthetic images composed entirely of Gaussian noise. The noise is generated to mimic the color distribution of the CIFAR-10 dataset, using specific means and standard deviations for the RGB channels. This extreme form of perturbation serves as a robust test of the model's ability to distinguish meaningful patterns from random noise, as it replaces the input data with purely random pixel values that follow the statistical properties of the dataset's color channels.

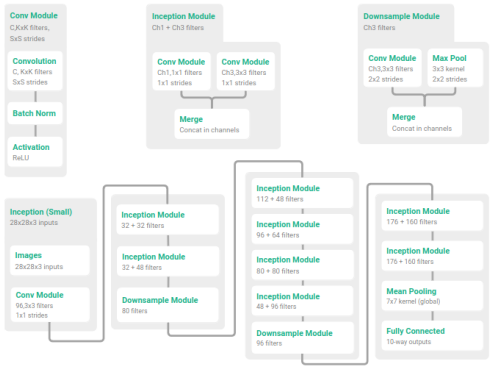

Fig. 2. Enter Caption

III. Used models: Inception Neural Network

A. Inception Neural Network

The first model used to carry out the different experiments with the CIFAR10 dataset was a small version of Inception. Inception is a deep convolutional neural network (CNN) designed for image classification tasks. The key innovation of the Inception architecture is the use of "inception modules", which are blocks of layers that perform multiple convolutions of different sizes and concatenate their results. This allows the network to capture features at multiple scales efficiently.

Using PyTorch was possible build Inception Architecture that is briefly explained in "Fig. 2". The base module is a ConvModule, it defines a basic convolutional module consisting of a convolutional layer, batch normalization, and ReLU activation. In the convolutional layer are defined the basics parameters, there are *in-channels* which represents the number of channels in the input data, typically corresponding to color information in images (RGB). *out-channels* determines the number of filters or feature maps produced by the convolution operation, each capturing different aspects of the input data. *kernel-size* specifies the dimensions of the convolutional filter, controlling the spatial extent of features that the filter can detect. *stride* dictates the step size at which the filter moves across the input data, influencing the spatial dimensions of the output. *padding* involves adding extra border pixels to the input, helping preserve spatial information during convolution and ensuring the output dimensions match the input.

This building block is then employed within the InceptionModule, where multiple branches conduct convolutions of varying sizes (1X1 and 3X3), attending feature extraction at different scales. Through concatenating the outputs of these branches, the module enriches the model representation capacity. InceptionNet further integrates DownsampleModules,

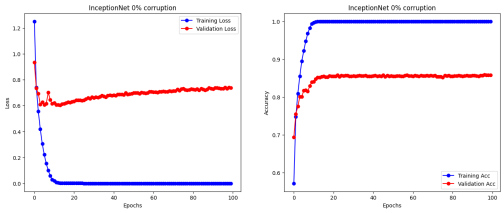

Fig. 3. Results for true labels

which facilitate spatial reduction through convolution and max pooling operations.

As the network progresses, successive layers of Inception and downsampling modules iteratively distill intricate visual features, taking advantange of both local and global context to comprehend image content comprehensively. This hierarchical feature extraction, culminating in a linear classifier, enables InceptionNet to discern diverse visual concepts with remarkable accuracy. At its core, InceptionNet embodies a sophisticated architecture adept at discerning intricate patterns within image data. By employing Inception and downsampling modules, the network efficiently captures features across multiple scales while mitigating computational complexity.

IV. Results

The experiment would involve training the Inception model on multiple versions of the CIFAR-10 dataset, each modified by different types of noise—label corruption, shuffled pixels, random pixels, and Gaussian noise. The goal would be to assess how well the model can learn meaningful features and maintain high accuracy despite the introduced noise. ”Fig. 3” to ”Fig. 5” show the results using different percentages of corruption, as can be seen in the graphs, at the beginning of the experiment without any noise the model is able to train in 100 epochs until reaching a training loss very close to zero, blue line, this is the normal procedure that happens with different neural network models where an iterative process is performed until reaching a threshold with the training data. On the right you can also see the accuracy of the model for each training and validation dataset, as seen in the graph with the clean data the model reaches a perfect accuracy very close to 1 in the training set, while with the validation set the accuracy is at 0.8 and 0.9, it is concluded that the model is useful and can be used to make predictions. From the following images we apply noise at different levels of corruption and as seen in the images, the training loss always reaches zero indicating that our model has enough capacity always to even fit even if it is only noise in the dataset.

The experiment was run for 100 epochs for each of the datasets, and as can be seen in ”Fig. 9” the we have that a close training loss is always reached, for Random Pixels and Gausian Noise, although it clearly does not reach zero, it is assumed that due to the inability of techniques to train the model for more than 100 epochs this result cannot be visualized, however, reference is made to ”Fig. 10” taken from

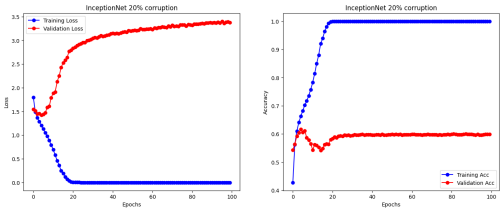

Fig. 4. Results for 20% corruption

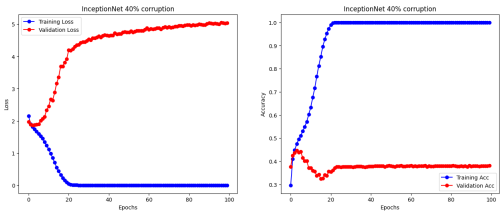

Fig. 5. Results for 40% corruption

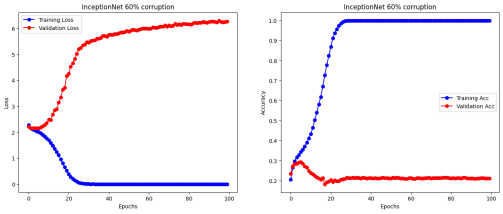

Fig. 6. Results for 60% corruption

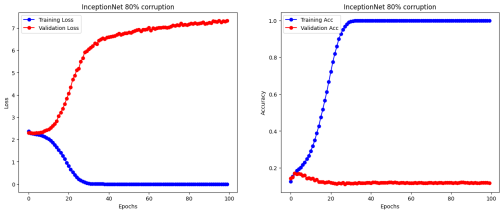

Fig. 7. Results for 80% corruption

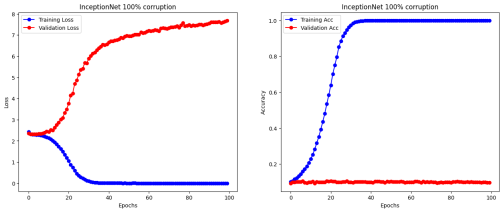

Fig. 8. Results for random labels

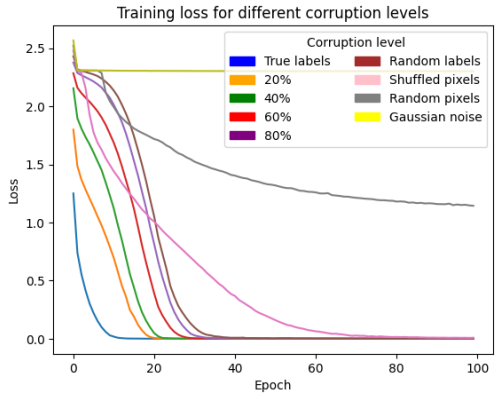

Fig. 9. Training loss for different noise data

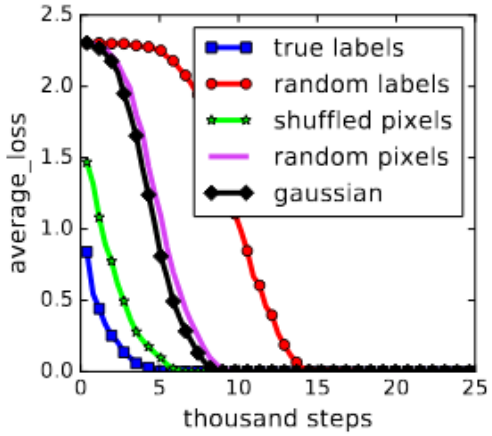

Fig. 10. Avg loss original experiment

the original experiment where with sufficient technical capabilities up to 25 thousand epochs can be run. The capacity of deep neural networks to fit random data, revealing significant insights into their learning abilities, was conducted experiments where they trained standard neural network architectures on datasets with randomly shuffled labels. Remarkably, the networks achieved zero training error, indicating their ability to memorize even completely random labels. However, the test error remained at random chance levels, as expected, due to the lack of correlation between training and test labels. This finding underscores that neural networks possess a high effective capacity, allowing them to memorize entire datasets, and that optimization remains relatively unaffected by label randomization. Extending this, the study also replaced images with random pixels (like Gaussian noise) and observed that convolutional neural networks could still fit this data with zero training error, at least in the original experiment. Further highlighting their ability to memorize noise. As the level of noise increased, generalization performance steadily deteriorated, suggesting that while neural networks can capture any remaining signal, they also brute-force fit the noisy data.

REFERENCES

[1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, “Understanding Deep Learning (Still) Requires Rethinking Generalization”, March 2021.

[2] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals, “Understanding Deep Learning Requires Rethinking Generalization”, February 2017.