Understanding Regularization and Why Regularization is Important in Neural Networks?

By: Kadali Manoj Teja (Student ID: 23017232)

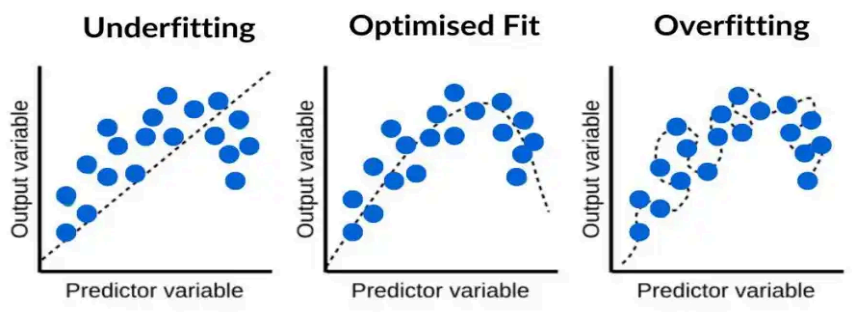
Introduction:

Regulation refers to the injection of constraints or penalties to a model to napkin it well to those data that do not belong to that model. Think of it as a controlled discipline, having a push from the neural network toward simple or uncomplicated modeling rather than toward complex modeling.

How Regularization is Essential for Neural Networks:

Regularization is the secret ingredient that forbids neural networks from "memorizing" data rather than "learning" from it. Aiming for the neural network to generalize: that is the goal when training it on data. In practice, the model ends up doing the opposite called overfitting. It does very well on training data but poorly on anything else. Here comes the regularization; it enables reinforcement so that the model does not needlessly complicate its understanding.

This entire tutorial serves to enlighten what regularization is, why it is so essential, and how everybody's diversity of various regularization techniques attracts or repels a model. All these are quite demonstrated using Python and PyTorch on the MNIST data set.



Overfitting, optimized fit, and Underfitting of a model

As shown in the graph, we can see that there are cases where the model is over-fitted and under-fitted but with regularization, the model has the optimized fit. The optimized fitting can reduce the accuracy of the model but will help to prevent the over and underfitting of the model.

Regularization is crucial because of the following reasons:
Without employing any kind of regularization, neural networks may tend to:

- **Overfit:** Learning the noise in its training data.
- **Generalize poorly:** Show poor performance on data that it has never seen before.
- **Be complex:** Use unnecessary parameters slowing down the predicting speed.

Now let’s learn how regularization works with an example from our daily lives,

Let’s consider that you teach a child to recognize dogs. Show pictures to a child: dogs and cats, so that the child learns to memorize every minute detail, such as the color of fur, the exact shape of ears, and so on. It passes with that picture but doesn't work with a new picture of a dog. It becomes confusing as it doesn't match with memorized details. This phenomenon is called overfitting-the child memorizes instead of understanding the general concept of 'dog'.

Now, to make that learning true, you do two things:

That is, you limit unnecessary detail: Ignoring the color of the fur or the background of the photo, you ask the child to recognize the dog through more general features like size, shape, and tail presence.
Encourage broad thinking: Give them really hard exercises, like trying to recognize a dog from a blurry image. Then they'll stop thinking based on exact details.
That's it; this is what regularization does in machine learning. It keeps the model from memorizing the training data and forces it to learn general features that work well on new data.

Types of Regularizations:

Regularization can be applied to a neural network using various techniques and some of them are given below;

1. Regularization L1 and L2

L1 Regularization (Lasso):

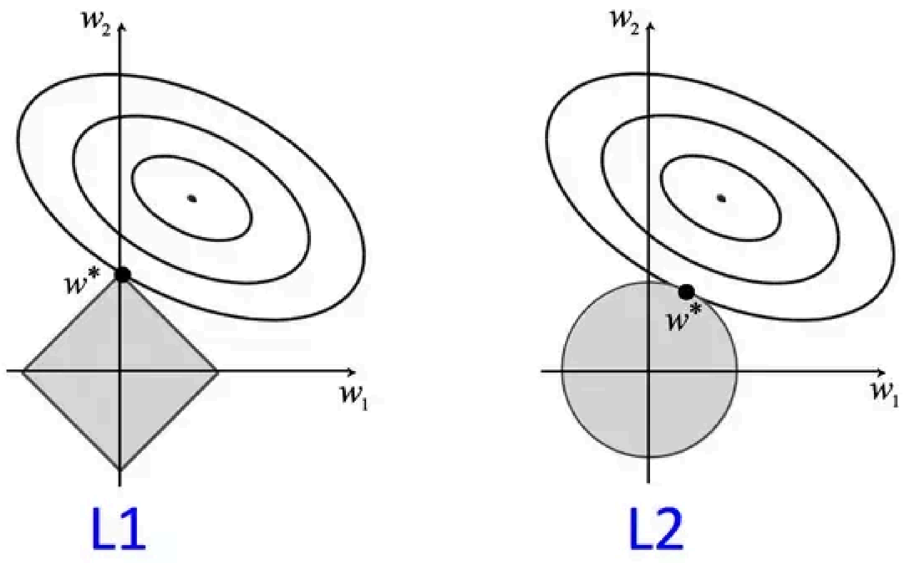
What it is: It's like a gate to the model. It forces the model to set some weights (the "attention" it gives to features) to be exactly zero. It is like telling to the model: "You do not need to pay attention to some features at all."
How: By adding a penalty to the extent to which the input weights are scaled, forcing completely ignoring certain features, leaving the model to focus only on the most important ones.
Effect: L1 regularization gives sparse models, meaning that it can eliminate some features totally. This can be quite helpful while you have a lot of features but only a handful of them can be real important.

L2 Regularization (Ridge):

What it is: L2 regularization is like telling a model, "Do not let any feature get too powerful." It does not make weights zero but shrinks all features by adjusting weights so that each of them has more balanced power in the model.
How: Penalty increases as weights grow larger. Encourage the model to have all features but in less amounts.
Effect: L2 keeps the smoothness of models and makes no one feature too dominating. It is good when you want all features to have influence but without very powerful domination of a single one.

Difference in L1 and L2:

L1 can give you sparse models with some features ignored completely, whereas L2 would give the smaller weights with all more balanced weights.

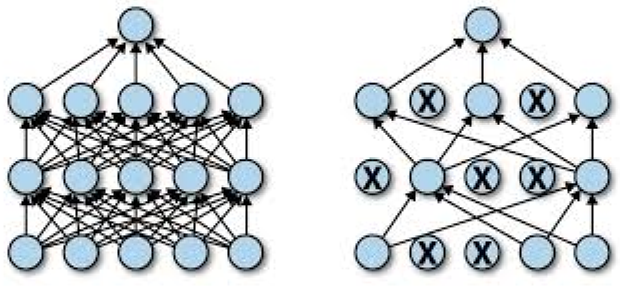


L1 and L2 regularization intuition

Thus, in practice, L1 would help in the event you wanted selective feature selection whereas L2 would primarily help in the event you wanted to keep almost all features when preventing overfitting.

2. Dropout:

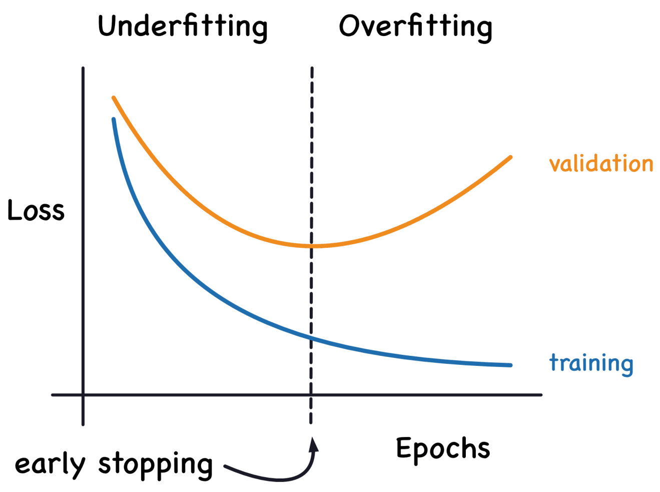
What it is: The dropout is a technique in which a certain percentage of weight during training of a neural network is randomly dropped or turned off. It is like a game where some of the players are blindfolded and then made to play the game, where they cannot rely on any specific teammate.
How it works: During every training step, a random set of neurons gets set to zero, so that it will not contribute to either forward or backward pass. This forces the network to learn redundant representations of the data because it cannot depend on the contribution of any single neuron for making predictions.


Dropout Regularization

Effect: Dropout prevents the model from becoming utterly reliant on specific neurons and helps in generalizing better to new data. It becomes particularly handy in deep networks, which usually become over-fitted.

3. Early Stopping

What it is: Early stopping is like stopping a race the moment a runner starts to slow down means not allowing the runner to finish. In machine learning terms, stopping model training once the model performance does not improve on the validation set without completing all the training epochs is referred to as early stopping.
How it works: The model performs constant evaluation on a validation set while being trained. The performance of that model on the validation set degrades (the loss starts to increase) at this point; it is an indication that the model has been overly specialized on what is in the training data and fails to learn generalizable patterns. The model is therefore over-fitted. Early stopping terminates the training process before the model overfits the training data.



Early Stopping Regularization

Effect: By means of this, the training can be wasted, and time can be saved by stopping the model once optimum performance on unseen data is reached. This prevents the model from wasting its resources and avoids overfitting by stopping at the optimum point.

Now let's compare a model with and without regularization and see how regularization affects the model:

We will use the Fashion MNIST dataset to train the models. First Normalize the data and scale the pixel values to the range 0 to 1.

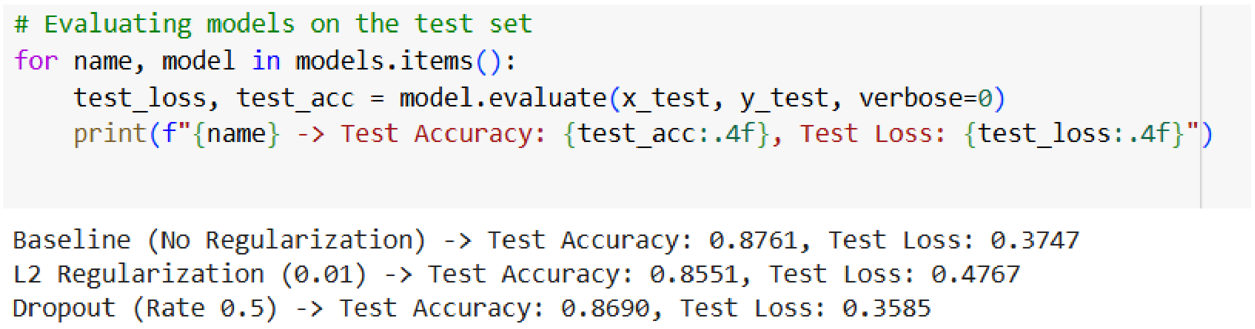
```
# Loading the Fashion MNIST dataset
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Normalizing the dataset (scaling pixel values to the range 0-1)
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Splitting of MNIST data into train and test data

Using a function that can create models with different regularization techniques we have created three models which are:

- Baseline (No Regularization)
- L2 Regularization (0.01)
- Dropout (Rate 0.5)

On training and validating these models on the Fashion MNIST dataset, we got the following results;



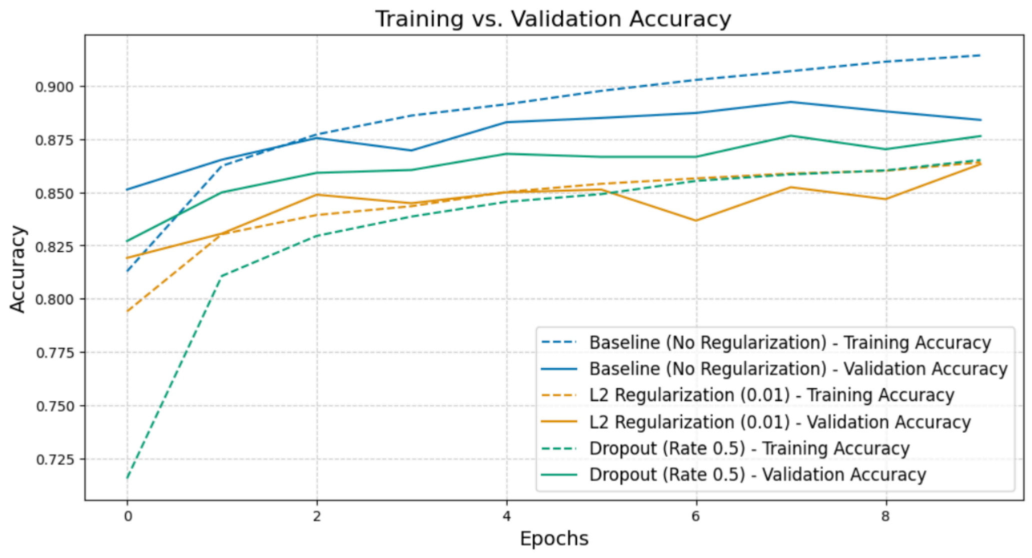
```
# Evaluating models on the test set
for name, model in models.items():
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    print(f"{name} -> Test Accuracy: {test_acc:.4f}, Test Loss: {test_loss:.4f}")
```

```
Baseline (No Regularization) -> Test Accuracy: 0.8761, Test Loss: 0.3747
L2 Regularization (0.01) -> Test Accuracy: 0.8551, Test Loss: 0.4767
Dropout (Rate 0.5) -> Test Accuracy: 0.8690, Test Loss: 0.3585
```

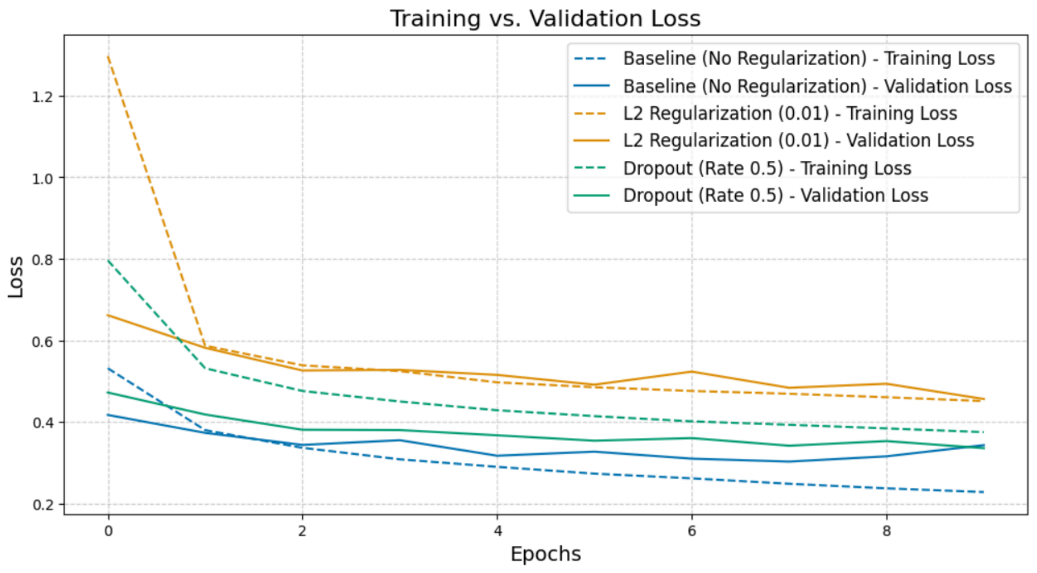
Test Accuracy and Test Loss of the three models

- The baseline model puts up the most accurate predictions on the test without regularization but is then susceptible to overfitting the training data. While the model does well on the test, overfitting denotes that it might not be able to generalize on outlier data found in real-world applications.
- The penalty on big weights-inhibiting L2 regularization caused the model to distribute learning over all neurons. This reduces the overfitting and increases generalization at a small added cost to baseline accuracy and added test loss. So, L2 regularization entails that some performance will be sacrificed to increase the robustness of the model.
- There is a good trade-off in dropout between accuracy and generalization: dropout allows flexibility and adaptability of models by keeping different neurons inactive during learning.

While comparing the models over training and validating accuracy and losses we have the following result,



Line graph comparing Training vs Validation Accuracy



Line graph comparing Training vs Validation Loss

Accuracy:

Baseline (No Regularization): The model tends to overfit with a high difference in accuracies measured during training and validation.

L2 Regularization: L2 regularization helps to mitigate the overfitting effect, as shown in the graph the gap is very low as compared with no regularization.

Dropout: Improvement through dropout causes increases in both training and validation accuracies.

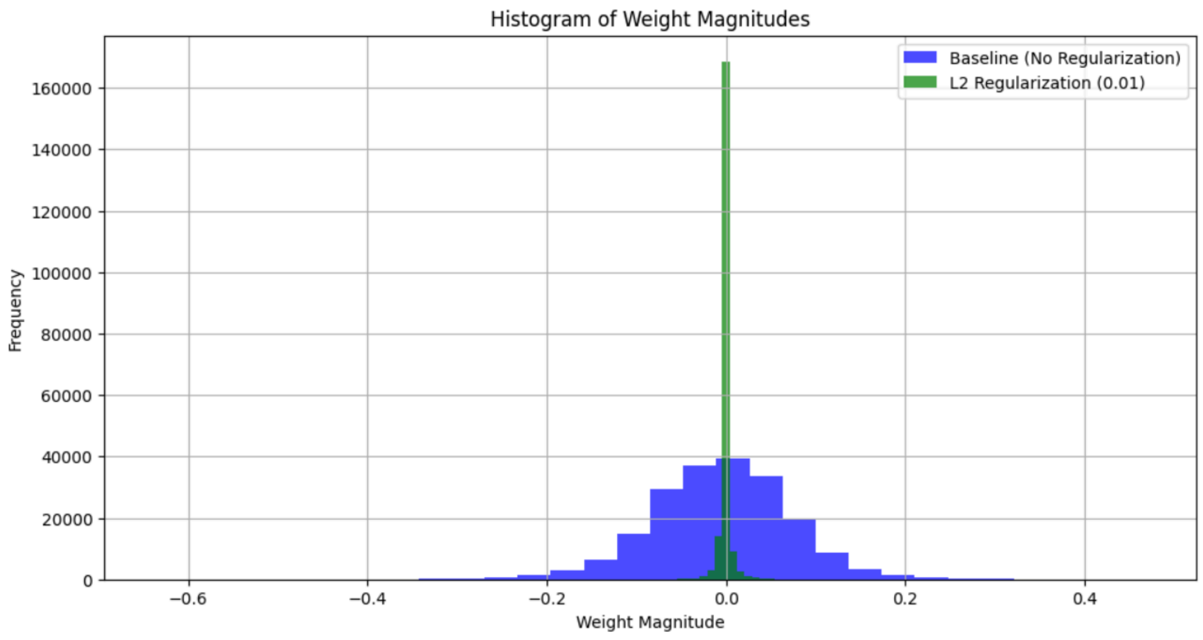
Loss:

Baseline (No Regularization): The model overfit as the training loss decreases rapidly but at the same time the validation loss increases.

L2 Regularization: The effect of the L2 regularization is to nullify the overfitting that results in a reduction in both training and validation loss.

Dropout: Further improvement of performance is brought by dropout low-loss training and validation.

Weight distribution is also an important part of using regularization as how the weights are distributed we can understand whether the model is memorizing or learning. On comparing the model without regularization and the model with L1 regularization we have the following result,



Histogram showing the Distribution of Weights of the two models

Observations:

No Regularization: Histogram depicts a broad distribution of weights, indicating that the model has learned complex features associated with very high weights; thus, this could lead to overfitting.

L2 Regularization (0.01): Notice that the histogram is less widened, more centered around the zero. This would suggest stopping L2 regularization from penalties for very large weights while making simple models that are less likely to overfit.

Conclusion:

Regularization techniques shine in increasing the neural network generalization ability, as this comparison reflects. The baseline model performs quite well in training; nonetheless, it carries this data over and shows a steep drop in accuracy with the validation. L2 Regularization uses a balanced method; it lowers training accuracy slightly but brings better performance in validation, thus pretty effective for moderations in overfitting. Among the others, dropout becomes the toughest tool for regularization, which indeed built a higher strength model along with the smallest amount of overfitting, as indicated by keeping training and validation accuracies more connected.

Though determining the correct one depends on your context, L2 regularization works for linear models or the kind of mild overfitting and dropout does for deeper neuromorphic nets where overfitting is witnessed. Applying those to your workflow would make your model perform better on unseen data with accuracy and reliability.

References:

[1] "Regularization in Machine Learning." *GeeksforGeeks*, GeeksforGeeks, https://www.geeksforgeeks.org/regularization-in-machine-learning/. Accessed 8 Dec. 2024.

[2] Shizuya, Yuki. "Understanding L1 and L2 Regularization with Analytical and Probabilistic Views." *Medium*, 13 Aug. 2020, https://medium.com/intuition/understanding-l1-and-l2-regularization-with-analytical-and-probabilistic-views-8386285210fc. Accessed 8 Dec. 2024.

[3] scikit-learn. "Underfitting vs. Overfitting." *scikit-learn*, https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html Accessed 8 Dec. 2024.

[4] Manfredi, V. "Lecture 12: Regularization." *Wesleyan University*,https://vumanfredi.wescreates.wesleyan.edu/teaching/comp411-f21/lectures/lec12-regularization.pdf Accessed 8 Dec. 2024.

Github Link for Code: https://github.com/DeepAstrix/Machine-learning-tutorial.git