Evaluation of regularization techniques on a simple Neural Network architecture trained on the Balanced EMNIST dataset

Federico Arenas Lopez

Abstract

During this study we will explore the different regularisation methods that can be used to address the problem of overfitting in a given Neural Network architecture, using the balanced EMNIST dataset. We will first identify the problem by varying the number of Hidden Units (width) and the number of Hidden Layers (depth) of the network, in order to understand the impact of overfitting for different Network Shapes. Subsequently, we will run different experiments to understand how to solve the initial problem adding Dropout, L1 and L2 Regularisation to a Baseline Model of fixed architecture and hyperparameters. These experiments will all be compared to the Baseline Model in order to find the regularisation method that provides the Baseline Model with the highest Accuracy and lowest overfitting.

1. Introduction

This study focuses on addressing the problem of overfitting by using explicit methods of regularisation such as Dropout, L1 and L2 regularisation on a Baseline Network. This problem will be addressed by running multiple experiments that will help us to understand the impact of these regularisation methods in a Baseline Network with a fixed architecture. By adding regularisation methods to an already overfitted Baseline Network with fixed parameters, we will see if there is a set of regularisation hyperparameters that are able to reduce overfitting and improve the model's performance.

All of these experiments will be done on the benchmark balanced EMNIST dataset (Cohen et al., 2017). This dataset contains 131,600 28x28 px images of 47 different handwritten digits. The training set counts with 100,000 images, the validation set counts 15,800 images, and the test set counts with 15,800 images. At each experiment, the networks will be fed the training set, validated and fine-tuned in the validation set, and the best network will be evaluated on the test set.

2. Problem identification

Overfitting happens when the Neural Network Model "learns by heart" the training data, and isn't capable of extrapolating what it learns to unseen data. Figure 1 in the coursework specs (Annex A) shows the performance during training of a Network for 100 Epochs, using Stocahstic Gradient Descent with a mini-batch size of 100, with Affine Layers followed by ReLu activations on all Hidden Layers, and all biases and weights initialised to 0. The network counts with the following architectural specifications:

Hidden Layers	Hidden Units	Learning Rate	Learning Rule
1	100	0.001	Adam

Table 1: Network Architecture and Learning Parameters for the Initial Network

From the figure, we can identify overfitting by looking at the performance of the model on the unseen data: the validation set. From the figure we can see that the Error Function starts to augment after around 15 epochs. This means that the model isn't able to learn new, unseen features. This translates into the accuracy on the validation hitting a maximum and starting to drop after 15 epochs. After 15 epochs, the model's generalization performance starts to drop. On the contrary, the performance on the training set continues to improve constantly. This will always be true for a model without any kind of regularization. From (Zhang et al., 2017) we know that during training, any model will be able to continue increasing its performance, until it has learned all the features of the training data.

2.1. Varying the number of Hidden Units and Hidden Layers

The previously identified problem leads us to begin testing different model architectures to better understand the presence of overfitting on different Network architectures. We stick to the Initial Model's Learning Parameters, but vary the architecture, starting by its number of Hidden Units.

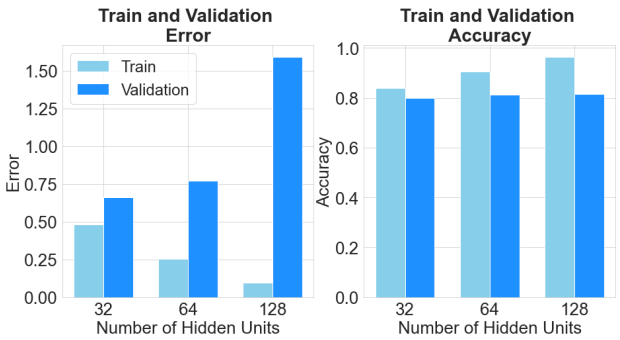


Figure 1: Initial Model's Performance for different Hidden Units

From the figure above we can see that, for a single Hidden Layer Network, the training error decreases and the training accuracy increases as the number of Hidden Units increases.

On the contrary, the validation performance shows exactly the opposite trend. This is because, as the number of Hidden Units increases, the model's ability to overfit to the data increases. However, because of the single layered architecture, the model isn't capable of learning "rules" that allow it to better generalize to unseen data. This insight leads us to test the same model with 100 Hidden Units for a varying number of Hidden Layers.

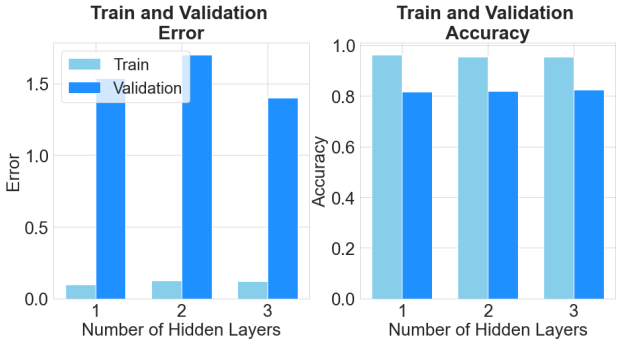


Figure 2: Initial Model's Performance for different Hidden Layers

The results from the figure above allow us to conclude that by increasing the number of layers of the model, we are giving it even more flexibility to learn the data by heart. This results in a very low training error and a very high validation error, which translates into overfitting. And also a large gap between the training and validation accuracies, which also translates into overfitting.

3. Dropout and Weight Penalty

3.1. Dropout Regularisation

The first method to address this problem is Dropout regularization. This technique, at each mini-batch, will shut off randomly selected hidden units in a given layer, be it an input layer or a hidden layer. Dropout does this by first assigning a probability (sampled from a uniform distribution) to each unit in a layer . Secondly, Dropout deletes all units that are under a given *Inclusion Probability* p , along with all of its incoming and outgoing connections to/from other units (Srivastava, 2014). The new, dropped-out layer will be fed into forward and backward propagation, and the same will be done for the rest of the layers of the network.

Finally, since the data has been reduced by a factor of p, we either re-scale during training or by using "inverted dropout" (dividing the number of units by p), or we multiply the final network by a factor of p. The algorithm for this method using "inverted dropout" is shown at the end of this subsection.

By running this algorithm through the entire network, we are creating an exponential number of networks that randomly differ from each other, and averaging their results. This allows to get an average network that has fewer units, and thus less room for variance to overfit to the data. Additionally, this allows for more "self-reliant units", that don't rely in other connections to make strong predictions, which in turn allows the network to generalize better to new data.

Algorithm 1 Dropout

Input: Inclusion probability p,
activations a^m_{l-1}, size mini-batch m, a_{l-1}
repeat
  Create uniform distribution U_d of size a_{l-1}
  for $i = 1$ **to** $m - 1$ **do**
    Assign U_d to a^i_{l-1}
    if $p > a^i_{l-1}$ **then**
      Keep a^i_{l-1}
    else
      Drop a^i_{l-1}
    end if
    Re-scale a^m_{l-1} by factor of $1/p$
  end for
until There are no more layers with dropout

3.2. L1 and L2 Regularisation

These techniques differ from Dropout because their goal is not to zero-out units in order to reduce variance, but to reduce the impact of units that count with very high weights. This is done by penalizing the error function E^m for a mini-batch m proportionately to how high the weights from a given layer are.

In L1 regularisation this penalization to the error function E^m is done by adding it the term $\beta|w^l_i|$, in L2 regularisation, it is done by adding the term $\beta\frac{1}{2}\sum_i w^2_i$

$$L1 : E^m = E^m_{train} + \beta|w^l_i|$$

$$L2 : E^m = E^m_{train} + \beta\frac{1}{2}\sum_{i=1}^{l} w^2_i$$

Where $|w^l_i|$ is the absolute sum of all weights in the given layer, $\frac{1}{2}\sum_{i=1}^{l} w^2_i$ is the L2 norm of all the weights in a given layer, and β is a hyperparameter that modulates how much weight penalty, and thus how much complexity/variance we want in the given layer. This penalization is also translated into backpropagation, taking the derivative of $|w^l_i|$ and $\frac{1}{2}\sum_{i=1}^{l} w^2_i$ and into account

$$L1 : \frac{\partial E^m}{\partial w_i} = \frac{\partial E^m_{train}}{\partial w_i} + \beta sign(w_i)$$

$$L2 : \frac{\partial E^m}{\partial w_i} = \frac{\partial E^m_{train}}{\partial w_i} + \beta w_i$$

This will allow β to modulate how much the weights are actually updated during backpropagation.

In L1 Regularisation the weight penalty will be done at a constant rate, whereas for L2 Regularisation we have a penalization that is proportional to the size of the weights. This means that when the absolute sum of the weights are high, L2 to will shrink the weights faster than L1. And

when the absolute sum of the weights is low, L1 will shrink the weights faster than L2.

4. Balanced EMNIST Experiments

Now we are going to test these regularization techniques on a Baseline Model.

4.1. Baseline Model

The Baseline Model will allow us to firstly define a fixed architecture on top of which the previously mentioned regularisation methods will be tested, and ultimately find the optimal set of hyperparameters that provide the best improvement of the Baseline Network. The Baseline Model has the Architecture and Performance shown in the table below.

Architecture			
Hidden Layers	Hidden Units	Learning Rate	Learning Rule
3	128	0.001	Adam
Performance			
Train Error	Valid. Error	Train Acc.	Valid Acc.
0.13	1.45	95%	82%

Table 2: Baselne Model's Architecture and Performance

From the table we can see that the model is overfitted. It counts with a 1.32 difference between the Training Error and the Validation Error. Additionally, the Network has a very high Training Accuracy compared to the Validation Accuracy. This translates into a low generalization performance.

The architecture from this Baseline Model will be kept for all new networks created further in the study. The same activation functions and weight and biases initialisation from §2 will be kept. This allows us to reduce the hyperparameter search space and focus on studying the direct impact of regularisation techiniques and its hyperparameter variations on increasing the generalization performance, and the general performance of the Baseline Network.

4.2. Analyzing the impact of Dropout on the Baseline model

Firstly, we study the impact of Dropout regularisation on the Baseline Model by training 3 different Models with the Dropout Hyperparameters specified in the table below.

Model	Incl. Probability	On Layer		
1	0.2	2/3/4		
2	0.5	2/3/4		
3	0.8/0.5/0.2	2/3/4		
Model	Train Error	Valid. Error	Train Acc.	Valid. Acc.
1	2.45	2.48	27.4%	26.6%
2	0.98	1.05	69.6%	67.9%
3	0.92	1.45	71.1%	68.8%
Baseline	0.13	1.45	95%	82.4%

Table 3: Dropout Models with their respective regularisation hyperparameters and their performances

As expected, Model 1 drops too many Hidden Units, which stiffles the learning of the network and doesn't allow it to converge to a lower error. As for Model 2 and 3, both count with very similar performances and although not better than the Baseline Model, they significantly reduce overfitting in their resulting networks. This can be seen in the figure below, where the barplots for Train and Validation have very similar heights.

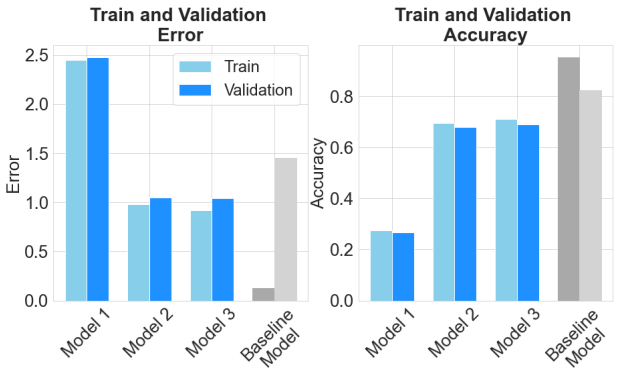


Figure 3: Performance of the 3 Dropout Models

4.3. Analyzing the impact of L1 and L2 Regularisation on the Baseline model

Now we want to analyse the impact of L1 and L2 Regularisation on the Baseline Model's performance. To do this, we train 3 L1 Regularisation Models and 3 L2 Regularisation Models, with the same hyperparameters in order to compare their performances.

Model	Regularisation	Weight Penalty	On Layer	
1	L1	1e-4	1/2/3/4/5	
2	L1	1e-2	1/2/3/4/5	
3	L1	Increasing WP	1/2/3/4/5	
4	L2	1e-4	1/2/3/4/5	
5	L2	1e-2	1/2/3/4/5	
6	L2	Increasing WP	1/2/3/4/5	
Model	Train Error	Valid. Error	Train Acc.	Valid. Acc.
1	0.33	0.43	88%	85.2%
2	3.85	3.85	2.2%	2%
3	3.85	3.85	2.2%	2%
4	0.15	0.67	94%	84%
5	0.77	0.79	77%	76.2%
6	0.44	0.52	85%	83%
Baseline	0.13	1.45	95%	82.4%

Table 4: L1 and L2 Regularisation Models' hyperparameters and their performances. *Increasing WP = 1e-4/1e-4/1e-2/1e-2/0.1*

Model 2 and 3 showed very bad performances. These performances may be due to a low absolute sum of the weights in the hidden units. This makes L1 regularisation shrink the weights faster than L2, and thus an L1 penalty of 1e-2 is high enough to shrink the weights by a high enough

rate that the learning is completely stifled and the function does not converge. Which would account for a very low accuracy, and a very high error.

Additionally, the results highlghted in green are the ones that (1) have the lowest gap between training and validation error, and training and validation accuracy. And (2) Count with the highest accuracy.

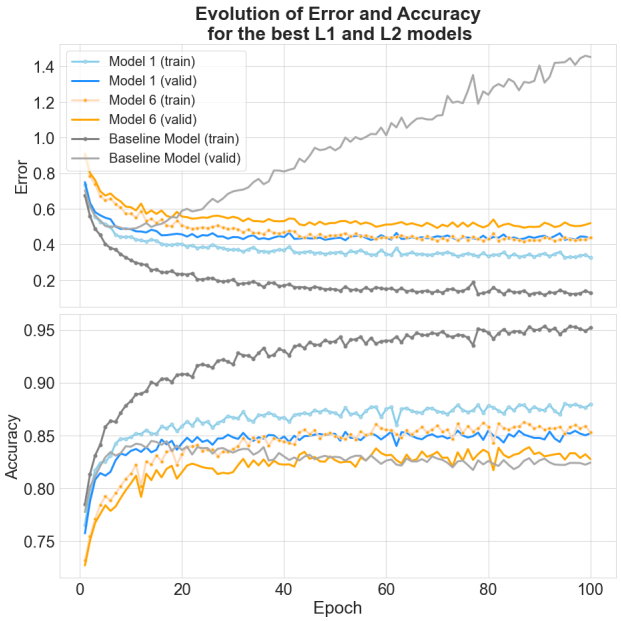


Figure 4: Evolution of the performance of the best L1 and L2 models compared to the Baseline Model

Finally, the figure above allows us to visualise how the Baseline Model starts to overfit after around 15 epochs. This problem is succesfully addressed by the best L1 and L2 regularisation model, where a stable local optimum is found after 40 epochs.

4.4. Analyzing the impact of Dropout and L1 Regularisation on the Baseline model

Now we are going to study the impact of combining Dropout with L1 Regularisation by modulating the Inclusion probability and the Weight Penalty in certain layers. The table below shows the different models tested and their respective results, compared to the Baseline Model.

Model	Incl. Prob./Layer	WP/Layer
1	0.8/2-4	1e-2/1-5
2	0.5/2-4	1e-4/1-5
3	0.8/2, 0.7/3-4	1e-4/1-4, 0.1/5

Model	Train Error	Valid. Error	Train Acc.	Valid. Acc.
1	0.62	0.65	79.5%	78.8%
2	3.85	3.85	2.1%	2%
3	0.99	1.04	69.2%	67.8%
Baseline	0.13	1.45	95%	82.4%

Table 5: Hyperparameters for models with L1 Regularisation and Dropout and their respective performances. *WP = Weight Penalty*

In the following plot we can visualize Model 1 performing very well, and Model 2 performing very poorly. Model 2's performance can only lead us to conclude that the hidden layers were penalized too heavily by dropping half of them and applying a Weight Penalty on it. The model does not seem to have converged to a local optimum and thus is unable to make accurate predictions, which accounts for the very low 2% Validation Accuracy.

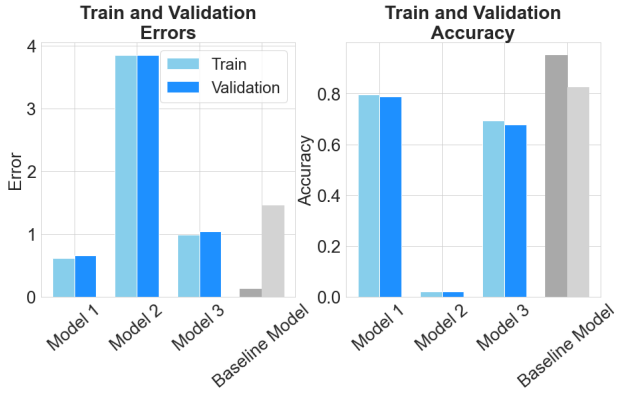


Figure 5: Barplot of performances from models with Dropout and L1 Regularisation compared to the Baseline Model

On the other hand, from Model 1 we can see that if we are not too severe in the amount of Hidden Units that we drop, we can achieve a good balance where we have the best generalization performance from all tested models in the study.

4.5. Reporting the results of the best model to the test set

Finally, we gather the best Models of each regularisation technique, and we obtain the table below, which outputs Model 1 from the L1 Regularisation study as the best performing model, to which we will refer as Final Model for the last part of the study.

Model	§ of Study	Regularisation	Train Acc.	Valid. Acc.
1	4.2	Dropout	69.6%	67.9%
1	4.3	L1	88%	85.2%
6	4.3	L2	85%	83%
6	4.4	Dropout+L1	79.5%	78.8%
Baseline	4.1	None	95.4%	81.4%

Table 6: Summary of the best results obtained for every regularisation technique studied in the experiments

Even though Model 6 from § 4.4 outputs the best generalization performance, the Final Model compromises some of its generalization performance with a highest accuracy. This leads us to evaluate this model to the heldout test set:

Model	Train Error	Test Error	Train Acc.	Test Acc.
Final Model	0.34	0.47	87.6%	84.03%
Baseline	0.12	1.54	95.4%	81.4%

Table 7: Final Model and Baseline Model test performance

5. Literature Review: Understanding Deep Learning Requires Rethinking Generalization

The chosen paper seeks to make the case that the current regularization techniques that are widely used to reduce overfitting are not sufficient to understand the underlying reason for a given Network to generalize well to unseen data.

To prove this claim, the author initially tests the CIFAR10, Alexnet and MLPs benchmark datasets on multiple random modifications of the labels and input images (e.g. random labels, shuffled pixel, random pixel, Gaussian noise, label corruption),. To complement this study, the author provides the theoretical proof that there will always exist a two-layer neural network with ReLu activations and 2n + d weights that can represent any function on a sample of size n in d dimensions. Which provides further backup to the results obtained above.

On the basis of these experiments, the author applies multiple regularisation techniques to the benchmark networks on the assumption that, according to the Rademacher Complexity, if these actually improved generalisation directly, the networks would struggle in converging to learn the labels during training. However, controversially, the Neural Networks with both explicit and implicit regularisation are still able to converge successfully to the data. This is a concrete proof of the initial claim, proving that the regularisation techniques are not helping the network reduce the Rademacher Complexity, and thus not making the model *actually generalize*. The results also show that even without regularization, the models still perform well on the original labels, which undermines the need to always use regularization techniques to improve generalisation performance. Indeed, the author suggests that implicit regularization techniques such as batch normalization or modifying the learning rule can provide better generalization performances, but are not exclusively necessary.

This leads the author to conclude that there is still a lot of research to be done to understand how a Deep Neural Network generalizes to unseen data because traditional measures of complexity fail to model this, and because according to the findings the ability of a network to converge is detached from its ability to generalize on new data.

Finally, on a critical last note, we would like to point out that the paper provides the reader with concrete insights to understand the complexity of improving generalisation of a certain Neural Network. Indeed, the paper shows that Neural Networks, because of their balck-box nature, have behaviors that are still difficult to understand, such as its ability to generalize. However, the author does this in a very unsystematic and disconnected way, letting the reader do the heavy lifting of deducing how the Rademacher Complexity, finite-sample expressivity, and training multiple networks on randomized data, are connected to generalisation, and thus are connected to regularization techniques. This is a very complex task. If the author could re-order the way these concepts are introduced, the paper would be much more clear.

6. Conclusions

The results from the study are satisfactory because we were able to, solely with regularisation, (1) greatly lower the Train/Test Error Gap from a 1.42 to a 0.13 Gap. Lower the Train/Test Accuracy Gap which went from a 14% to a 3.57% Accuracy Gap. Additionally, (2) we were able to increase the Test Accuracy from 81.4% to 84.03%. The Final Model is able to stably converge to a local minimum after 15 Epochs of training:

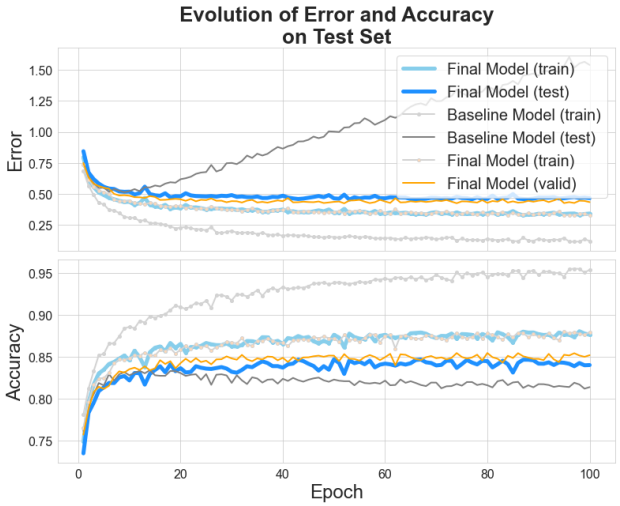
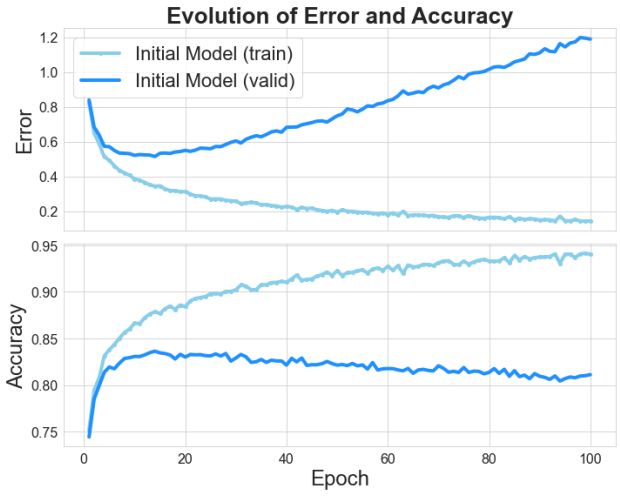


Figure 6: Performance of the Final Model on test set compared to its performance on the validation set

Finally, it would be interesting to have the same study done for implicit regularization techniques such as batch normalization, early stopping, and changing the learning method, (Neyshabur, 2017). This would be useful to better understand the nature of generalization in Neural Networks, and the challenge presented by (Zhang et al., 2017).

7. Annex A



Annex A: Performance of the Network Shown in Figure 1

of the coursework specs

References

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. 2017. URL http://arxiv.org/abs/1702.05373.

Chiyuan Zhang, Benjamin Recht, Samy Bengio, Moritz Hardt, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

Krizhevsky Sutskever Salakhutdinov Srivastava, Hinton. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 06 2014.

Behnam Neyshabur. Implicit regularization in deep learning. 09 2017.