Impact of L1 and L2 Penalties on Generalizability of a Deep Learning Model

Jordan D. Levy; University of California, San Diego

ABSTRACT

This study aims to address the affects of L1 and L2 regularization constraints on the ability of a deep learning model to generalize to unseen data. We look at the Tiny-ImageNet dataset containing 500 images each of 200 different classes. By using three different Convolutional Neural Network (CNN) architectures, referred to throughout the paper as the Initial Model, New Model, and the well-known ResNet, we assess the impact of L1 and L2 penalties on the training and validation accuracies of each model trained between 25 to 50 epochs, as well as the loss at each epoch. We conclude that the L1 penalty results in a loss curve that decreases with the training loss more-so than the L2 penalty. This may result in better validation accuracy, but the results are inconclusive. The architecture itself carries more importance to the validation accuracy, but regularization in conjunction with other hyper parameter scans has the potential to make a significant difference in the validation accuracy and generalizability of a given deep learning model.

1. INTRODUCTION

In any machine learning or deep learning project, the goal is to come up with a model that not only fits your training data very well, but is also able to generalize and generate good predictions on unseen data. While fitting your data ends up not being too difficult relatively speaking as we can turn most prediction tasks into a math problem and simply minimize our loss function, creating a model that can accurately predict unseen data is a much more challenging problem. Creating high-level abstractions of our training data to then predict unseen data is at the heart of

machine learning and especially deep learning.

In traditional machine learning, we aim to create a vector of weights **w** that fit to an input matrix **X** such that the predicted outcomes $\hat{\mathbf{y}}$ and the ground truth outcomes **y** have the least possible difference. However, one must consider the generalizability of such a model.

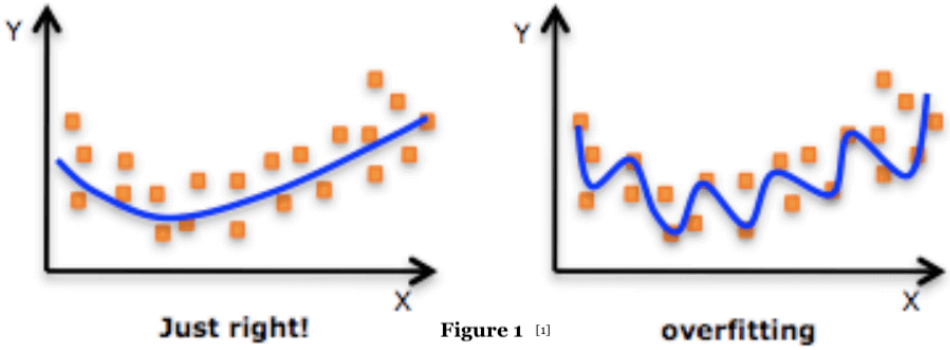


Figure 1 [1]

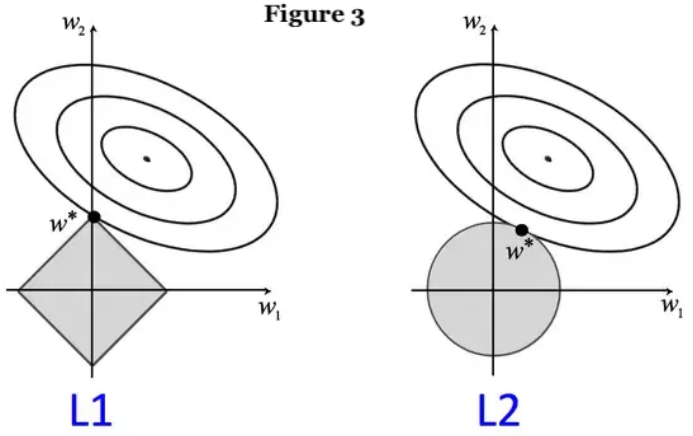
In Figure 1[1], we see the difference between two models with the one on the left capturing the overall trend of the data while the one on the right overfits the data. We expect that if the data on the right is a representative random sample of the population from which the data is gathered, then the model will do a pretty good job at predicting the values of unseen data, whereas the model on the right is overly sensitive to the noise in the dataset as it tries too hard to fit every training data point. So how do we prevent our model looking like the one on the right and more like the one on the left? In traditional machine learning, this problem is tackled through regularization penalties that are added to the loss function.

Two of the most common regularization penalties applied in machine learning are called L1 and L2 penalties. L1 penalty adds the absolute sum of all coefficients to the loss function, while L2 penalty adds the sum of square coefficients.

Figure 2 [2] shows us the L1 penalty added to the sum of squares loss function. Here we see L1 applied as hyper-parameter lambda times the absolute sum of coefficients.

In Figure 3 [3], we assess the different properties of L1 and L2 regularization. Essentially, we

Figure 3



see that L1 is a linear function between our weights and loss, $|\hat{\beta}_j|$. while L2 is quadratic. In both instances, the goal is to reduce the magnitude and/or number of parameters such that we get a smoother curve for predictions or decision boundary for classification. In Figure 1, the model on the right has more parameters and larger absolute coefficients for each parameter than the model on the left. We therefore hope to reduce the complexity of our model through L1 and L2 regularization, thus improving the generalizability as well.

Regularization in its entirety aims to tackle one of most important concepts in machine learning: the bias-variance tradeoff. When we regularize a model, we are allowing for more bias (error in predictions on the training set) in exchange for less variance, meaning less variability of predictions given a similar input. We hope that by slightly increasing our bias, we can create a more robust model that will give us higher quality predictions on unseen data.

While we have seen that regularization is a popular technique for tackling the bias-variance tradeoff in traditional machine learning, we want to see how this can be applied in deep learning. Specifically, we will be assessing the impact of L1 and L2 regularization on the training and validation accuracy on the Tiny-ImageNet dataset, a collection of 200 classes of 64x64 RGB images, with 500 pictures in each class. We will use three different Convolutional Neural Networks (CNNs) to approach this problem: two initial models with fairly simple architectures, and the

ResNet architecture. For each network, we will be adjusting lambda values for L1 and L2 regularization, with the hope of observing the effect of each regularizer on the generalizability of the model. The way we are assessing generalizability is through the highest validation accuracy found on the Tiny-ImageNet dataset after running our model for 25 epochs.

2.1 Initial Model Regularization

Our first step is to build a simple CNN to see what baseline accuracy we can build off of for Tiny-ImageNet.

Figure 4

```
Net(
  (conv1): Conv2d(3, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=5120, out_features=2000, bias=True)
  (fc2): Linear(in_features=2000, out_features=200, bias=True)
)
```

In Figure 4, we see our model is comprised of two convolutional layers, both with 3 by 3 kernels, with a 2x2 average pooling layer applied after each convolution. This is followed by two fully connected layers that reduce the dimensionality from 5120 to 200, thus outputting a softmax probability for each class. Using this relatively simple model, we get the following plot for train and validation loss/accuracy at each epoch when setting L1_lambda and L2_lambda to 0:

Figure 5

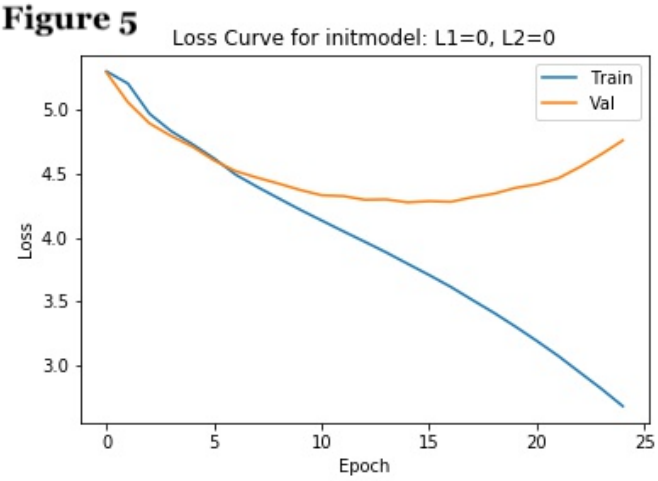
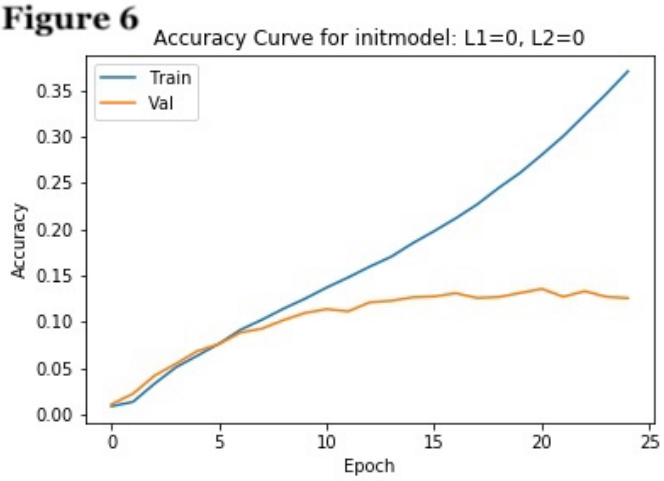


Figure 6



From Figures 5 and 6, we can see that the model starts to overfit sometime between epochs 10 to 15. This is because the validation loss curve begins to go back up after the 15th epoch and the training accuracy begins to diverge from the validation accuracy around the 10th epoch.

Next, we test the Initial Model with an L1 lambda of 0.0001 and L2 lambda of 0.

Figure 7

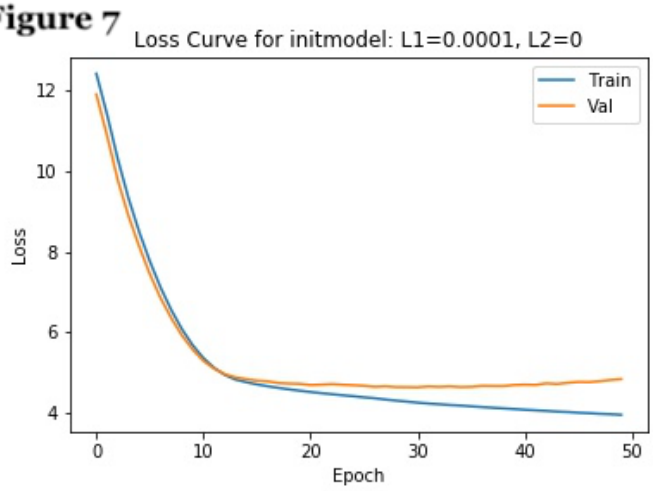
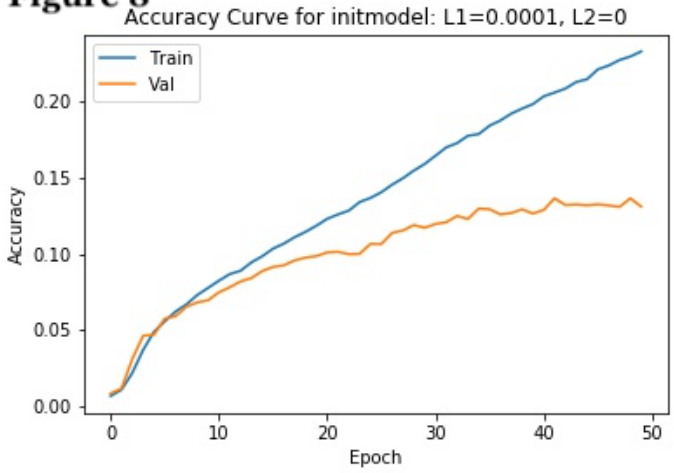


Figure 8



Figures 7 and 8 tell a much different story than Figures 5 and 6, both in the magnitude of the losses and in train and validation accuracy curves. Now that we have added an L1 penalty to our loss function, our validation loss and training loss appear to decrease at a similar rate through the first 15 or so epochs, with the training loss starting to diverge around the 20th epoch. By training with 50 epochs instead of 25 we start to see an increase in validation loss, but not to the same extent as in Figure 5. Additionally, we observe that the accuracy curves are much closer between training and validation sets in Figure 8 than they were in Figure 6, but the overall training accuracy is much lower in Figure 8 than in Figure 6 (0.14 to 0.36). This shows that by adding an L1 penalty to our loss function, we are effectively reducing the training accuracy seeming to increase the validation error over a longer training period. The differences between Figures 5, 6 and Figures 7, 8 seem to be a prime example of the bias-variance tradeoff, as we are reducing the

complexity of our model via the L1 penalty, reducing our variance which should increase our validation accuracy at the cost of reducing our training accuracy. Ultimately, however, our goal is to best predict unseen data, making the tradeoff beneficial to our model's primary objective.

Now that we have seen how the model reacts to adding an L1 penalty, we will see what changes when we add an L2 penalty with no L1.

Figure 9

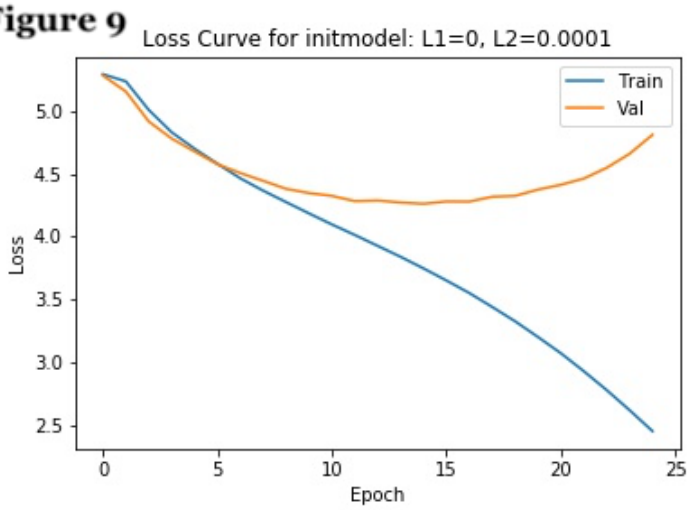
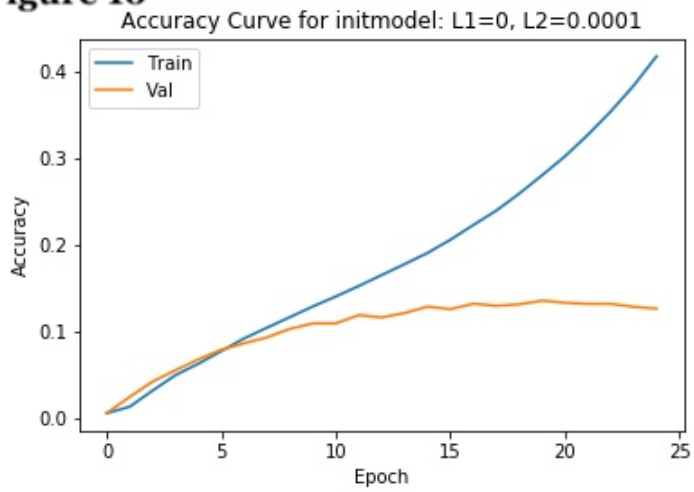


Figure 10



Interestingly enough, when we add L2 penalty with no L1, we see a similar trend as when we had no L1 or L2. The only difference between Figures 9,10 from Figures 5,6 appears to be the reduced training loss as well as increased training error. The training loss appears to decrease at a faster rate once the validation loss stops decreasing and starts to increase, suggesting that once the L2 penalty gets closer to 0 as the model's parameters get smaller, the model starts to overfit.

As a last check, we assess the model's training and validation accuracy when both an L1 and L2 penalty are applied, setting both lambdas to 0.0001. What we get is a similar loss and accuracy curve as Figures 7 and 8 but with a slightly higher training accuracy. This tells us that the L1 loss tends to slow down the rate at which the training accuracy increases, while the L2 loss increases that rate after a certain number of epochs. For those interested in seeing the actual

figures when L1 and L2 lambdas are 0.0001, follow this link that contains all figures produced for this report: **[4]**.

2.2 New Model Regularization

We have found the accuracy of our baseline model, we will now look into a slightly more advanced model to see if the results differ as a result of the increased complexity.

Figure 11

```
Net(
  (conv1): Conv2d(3, 10, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool1): AvgPool2d(kernel_size=2, stride=2, padding=0)
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(10, 20, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(20, 30, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (fc1): Linear(in_features=1920, out_features=1000, bias=True)
  (fc2): Linear(in_features=1000, out_features=200, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
)
```

Figure 11 shows the new model architecture. Its main difference from the Initial Model lies in its extra convolution layer, as well as including MaxPooling after the second and third convolution layers in addition to AveragePooling for the first convolution layer. Another change comes with the dropout layer, where nodes from the pooling are disconnected from the fully connected layer with 0.2 probability. Dropout is another method on top of regularization designed to improve generalizability. Looking at the case where L1 lambda and L2 lambda are both 0, we see a very similar trend to that of the Initial Model in Figures 5 and 6, but with a 2% bump in validation accuracy (13.6% —> 15.7%). We also see a similar trend for L1 lambda = 0.0001 and L1 lamb-

Figure 12

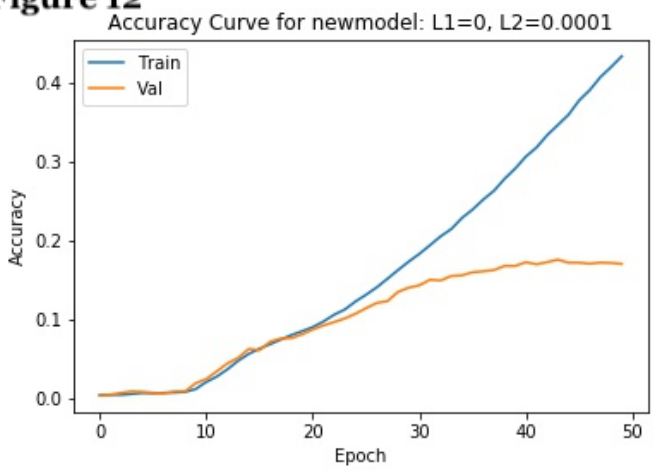
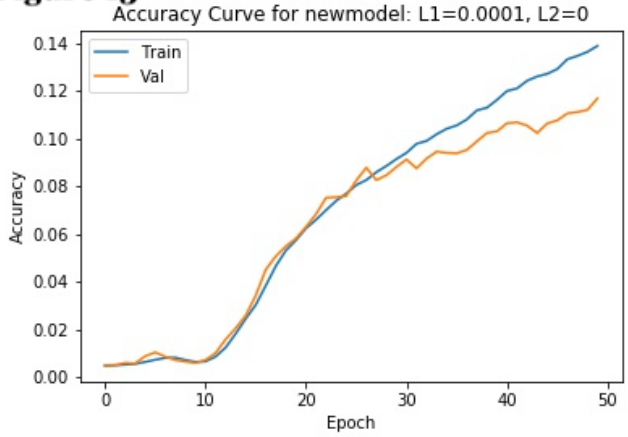


Figure 13



da = 0, but the New Model is a slower learner than the Initial Model. Ideally, we would want more epochs to see the effect more clearly, but L1 seems to outperform L2 again, with the combination of L1 and L2 giving worse results. Although the peak in Figure 12's validation accuracy, 0.1764, is considerably higher than the peak in Figure 13's validation accuracy, 0.1169, we cannot count out L1 for the New Model as the validation accuracy continues to grow in Figure 13 while it has become stagnant in Figure 12 by 50 epochs. We see a similar story with Figure 14 as with Figure 13, despite poor validation accuracy there is a lot of room for growth. It seems like the addition of L1 penalty leads to a delay in the learning process, something that should be looked more into to see why that could be the case.

2.3 ResNet Regularization

Now that we have looked at two custom designed CNNs and assessed the impact of L1 and L2 regularization on both, we look towards a mainstream architecture to see if the results will differ. We expect to get greater accuracy as ResNet is a tried and true CNN architecture. If you are not familiar with ResNet, the paper can be found here: **[5]**. We use this model with the only modification being the length of the output layer, from 1000 to 200 nodes. When looking at loss and accuracy curves for the same L1 and L2 lambda values used in the previous two model with ResNet, three apparent shapes form. Figure 14 describes the accuracy curve for the different values, tested, with the only difference being slightly less training accuracy (0.8) when more regularization Is introduced, as well as slight fluctuations of 1-2% in validation accuracy between the values

Figure 14

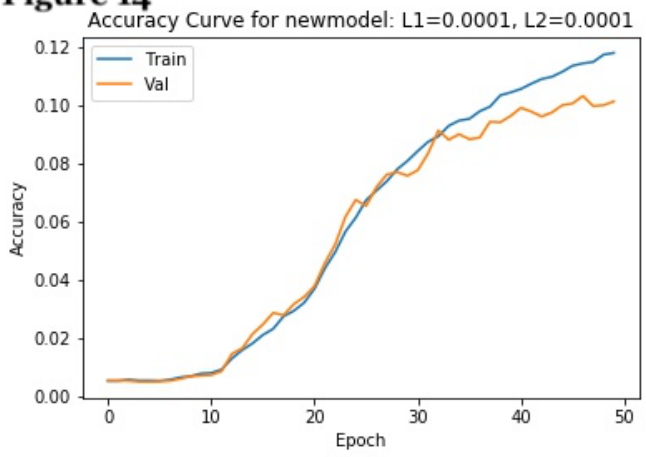


Figure 15

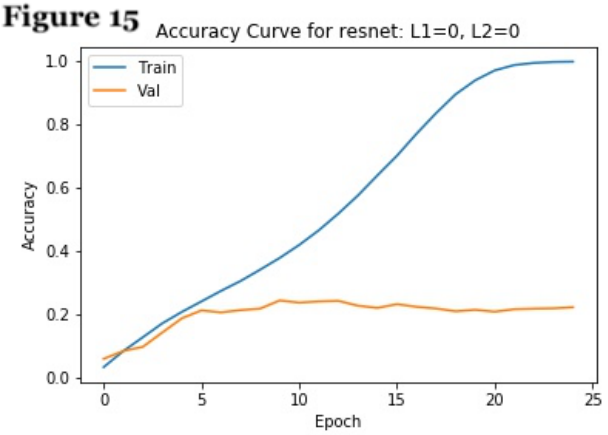


Figure 16

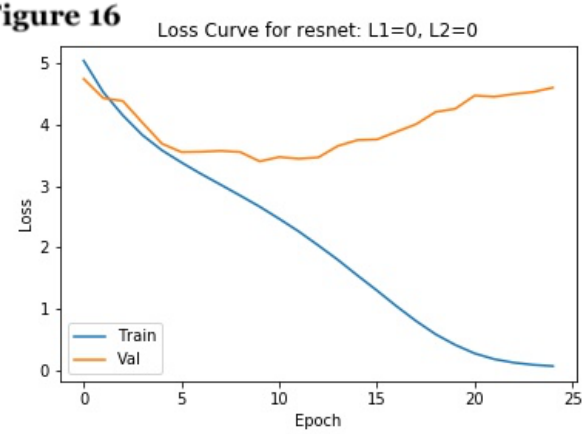
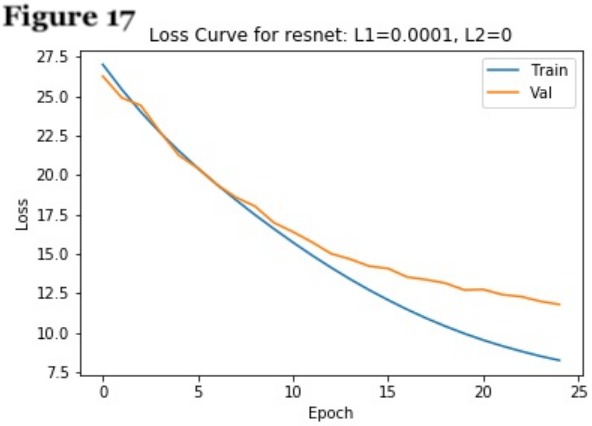


Figure 17



of lambda. The contrast between Figures 16 and 17 are quite profound, as we see validation loss continues to decrease at 25 epochs when L1 penalty is introduced, as opposed to increased after only 10 epochs with no regularization. Figure 16 also shows up when L2 penalty is introduced, suggesting yet again that L1 penalty has a greater influence on decreasing validation loss and possibly increasing validation accuracy. Overall, the main takeaways from ResNet are that it fits the training data much faster than the first two models, and gets about a 7-10% boost in validation accuracy from those models.

3. CONCLUSION

In the three tables below, we show the highest training accuracy and validation accuracy for each value of L1 lambda and L2 lambda, while specifying the model used and the epoch at which the best validation accuracy was reached [6].

Initial Model (50 epochs)	Train Accuracy	Best Validation Accuracy	Epoch of Best Validation
L1=0, L2=0	0.2907	0.1323	22
L1=0.0001, L2=0	0.2293	0.1364	49
L1=0, L2=0.0001	0.2408	0.136	19
L1=0.0001, L2=0.0001	0.1396	0.1116	25

Table 1

New Model (50 epochs)	Train Accuracy	Validation Accuracy	Epoch of Best Validation
L1=0, L2=0	0.2979	0.1571	38
L1=0.0001, L2=0	0.1389	0.1169	50
L1=0, L2=0.0001	0.3466	0.1764	44
L1=0.0001, L2=0.0001	0.1146	0.1034	47

Table 2

ResNet (25 epochs)	Train Accuracy	Validation Accuracy	Epoch of Best Validation
L1=0, L2=0	0.3776	0.2431	10
L1=0.0001, L2=0	0.4747	0.2559	13
L1=0, L2=0.0001	0.4615	0.2368	12
L1=0.0001, L2=0.0001	0.3959	0.2456	10

Table 3

Overall, we see that ResNet performed the best across the board, as was expected, but there are interesting trends to note within each table. The initial model has very slightly higher validation accuracy for L1 exclusively as opposed to L2 exclusively, but there seems to be room to grow as the best validation accuracy was reached at the 49th epoch out of 50. In the new model, this trend appears to be reversed, with the exclusive L2 model having almost 6% higher validation accuracy than the L1 exclusive model. However, there is room to grow in both cases as they reached their peak validation accuracy late in the training cycle. Interestingly, all of the ResNet models reached their peak validation accuracy within the first 15 epochs, suggesting that ResNet is quick to overfit the model, albeit providing superior generalization to the previous models.

Beyond the results found in this report, there are numerous ways in which this study could be extended and improved upon. While we can clearly show that the introduction of L1 and L2 penalties serves to decrease the training accuracy with the hope of increasing validation accuracy, we cannot conclusively make the claim that validation accuracy has increased with

regularization. However, the scope of this study was quite limited, we are only looking at one dataset and are neglecting other hyper parameters such as learning rate and momentum that may have adverse affects on our model's performance. These are all things that can and should be looked into in more detail.

A main goal of this study was to see how the most popular forms of regularization in machine learning affect the output of a deep learning model, and while the results were inconclusive and have raised more questions than answers, we are given a starting point with which to look into with greater depth. We hope to be able to more thoroughly answer the question of how L1 and L2 penalties affect deep learning models, where can improvements be made, and how or if these penalties can help explain the bias-variance tradeoff in deep learning the same way it does in traditional machine learning.

REFERENCES

[1] https://www.datarobot.com/wiki/overfitting/

[2] https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net

[3] https://codeburst.io/what-is-regularization-in-machine-learning-aed5a1c36590

[4] https://github.com/JordanLevy99/COGS_181_Final_Project

[5] https://arxiv.org/pdf/1512.03385.pdf

[6] https://docs.google.com/spreadsheets/d/1214GJFBk_lrrEvGlkenVSFTa-JG5ukq93OcHlhyxlgf8/edit?usp=sharing

[7] https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html