# Improving Neural Networks Through Optimization Techniques of Parameter Statistics

Joshua Salit

## Abstract

The goal of neural networks is to represent some complex relationship with data to generalize predictions across different inputs. Traditionally this is performed using simple evaluation metrics balancing overfitting and underfitting the data. The latent space representations created from neural networks doesn't always represent a simple linear relationship that can be comprehensively evaluated from two numbers. This analysis explores the use of weights, gradients, and activations to uncover important information about the neural network to inform constructing better neural network architectures.

## Introduction

Neural networks are designed to model complex tasks by generalizing latent space representations in data. Traditional methods for evaluating and approaching these models involve a simple comparison of overfitting and underfitting data. Modeling sophisticated representations with simple measurements seems like an insufficient method for achieving a holistic understanding of a network. This study attempts to discover other measurements to supplement the evaluation and optimization of neural network models.

The success in neural network models is evident in their widespread adoption over the years. These complex representations have become deeper over time with more and more layers and depth. Traditionally, evaluating these complex relationships and parameters to make predictions depend on two key numbers - a training and validation measurement. Many knobs exist in human's control and determining the optimal configuration isn't a simple task. Many

professionals employ systematic techniques such as grid search or algorithmic optimizations techniques to discover the best model. However, achieving a top of the line neural network contains a balance of art and science to achieve this success.

The goal of this study is to figure out what can we discover from the parameter of models to help influence human tuning in neural networks. In addition to weights, activations and gradients are explored and for simplicity collectively refer to them as parameters. The popular CIFAR-10 dataset is used as a relatively basic set of images to be predicted. Convolutional Neural Networks (CNN) is the architecture chosen to represent this data. This study was constructed based on three main sections- 1) run a multitude of models to gather data and calculate statistics, 2) analyze these statistics to figure out which are most important, and 3) determine what can be tuned by humans to achieve these optimal statistics. The assumption and hope is that achieving these optimal statistics will indirectly develop more powerful models.

Statistics calculated were based off of three primary parameters calculated as the neural networks were propagated - weights, activations, and gradients. Statistics calculated on these parameters were variance, maximum values, percent of the parameter below a threshold, number of parameters, and a couple other other calculations based on this data. The results were compared cross-sectionally and through time series as they traversed through each epoch.

Overall, the findings from this analysis can be summarized in a few general themes. A goal of implementing neural networks is to achieve low weight and activation variance, activation maximum values, and relatively low amount of parameters. This can be accomplished by focusing on the first couple of layers because they disproportionately have a greater effect on performance, and regularize the last few layers because these parameters can approach extreme values easier.  After discovering these results, my first model was able to achieve the

second highest results out of the models tested to obtain these numbers and eventually achieved the best model with a test accuracy of 74.7%, training accuracy of 96.3%, and validation accuracy of 75.6%.

**Literature review**

Discovering the most powerful model is always the task for anyone employing machine learning techniques. Over the past decade, an explosion of interest and success has been experienced in the neural network industry. Many of the “godfathers” of neural networks have had a sizable influence on this paper. Yann LeCun showed the prominice of CNNs in 1998 with the development of the LeNet-5 model representing a simple but effective architecture to lay the foundations of CNNs today (LeCun et al., 1998). Implementing neural networks with ReLU activation functions and with GPUs are both integral parts of this study to obtain fast results that converge quickly (Krizhevsky, Sutskever, Hinton, 2012). Bengio discusses foundational topics for tuning hyperparameters in neural networks exploring both systematic and non-systematic methods (Bengio, 2012). He develops generalized heuristics for optimizing models and also implements them with automatic approaches (Bengio, 2012).

Optimization of hyperparameters is always a focus as neural network models are built. The implementation of these tasks can be broken down into systematic and non-systematic approaches. Systematic approaches include grid search, random search, and other algorithmic methods to determine optimal parameter settings. More empirical research has been completed with these approaches rather than non-systematic. In the article by Bergstra and Bengio hyperparameter optimization is calculated based on a random search method that showed to perform better than grid-search (Bergstra and Bengio, 2012). A bayesian probabilistic approach is used to get appropriate settings showing some advantages (Klein et al., 2017). An example of

a more sophisticated approach is with the Tree of Parzen Estimators (TPE) algorithm demonstrating strong performance as well (Bergstra et al., 2013). Employing these systematic methods can help find the best model but understanding the underlying relationships driving these models is important. Augmenting this approach with a manual approach based on a foundational understanding of the dataset and model you are working with could provide an opportunity to improve performance.

**Data Preparation**

Transforming data serves as an important step to obtaining satisfactory results when training neural network models. The CIFAR-10 dataset is relatively simple, represented as 32 x 32 RGB pixel data with 50,000 training images and 10,000 test images. This dataset was developed by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton and the raw data was extracted from kaggle (Krizhevsky 2009). Validation data is created by taking 5,000 samples from the training data. Predicting 10 objects that are animals or machines is the goal of these experiments.

Transforming the data involved reshaping, scaling, and supplemental tasks to add to the models. Reshaping the data was necessary to provide the neural networks with appropriate shapes. To help the models converge easier, the images were down scaled to a range of 0 and 1 by dividing by 255, the max value. Slicing the data was implemented to be able to easily obtain batches.

After importing the raw data, the next step was to explore the data to discover nuances to be appropriately handled to model the data. Fortunately this simple dataset did not require substantial transformations. Maximum and minimum values were clearly cut off at the 255 and 0 thresholds. Null values were absent enabling the data to be easily inputted to networks. Data was shown by plotting the images showing clearly defined objects with good resolution.

**Research Design and Modeling Methods**

This analysis was constructed in three sections - 1) run a multitude of models to gather data and calculate statistics, 2) analyze these statistics to figure out the most predictive to obtain good performance, and 3) determine what can be tuned by humans to achieve optimal statistics.

Part one involved building the infrastructure to run models and collect all of the data needed to be analyzed. A flexible implementation was utilized with gradient tape enabling capturing and modifying the data relatively simple. The foundational infrastructure of this step was crucial for this study because all of the findings used this data collected. A couple steps contributed to his part of the study including building the neural network to run with gradient tape, collecting all of the parameter data, and calculating statistics based off of this data. Some of the models stored the parameter data however these arrays took up a lot of space so was not done for all models. A total of 57 models were run to gather the data. These models were relatively simple and only changed a handful of hyperparameter and model architecture knobs. This included a number of neurons and filters, number of layers, and the regularization techniques l1, l2, dropout, and batch normalization.

An integral part of this study was calculating statistics based off of the parameter data. The parameter data utilized was weights, activations, and variances. Statistics calculated on each of these parameters were variances, maximum values, and the number of weights below a threshold. Other statistics calculated were number of sparse activation values (ReLU activation function was used), number of parameters, and parameter efficiency. Parameter efficiency is simply the time to run a model divided by the number of parameters. This metric helped uncover the sometimes nonlinear relationship between number of parameters and performance of a model. These metrics were calculated after the training process and the data was stored.

Another important aspect of calculating the statistics was the temporal aspect of it by finding the relationship over time through each epoch. A cross-sectional and time series approach was taken when analyzing this data. Heatmaps were typically used to model the cross-sectional data whereas both line graphs and heatmaps were used to model the sequential data. Both revealed different perspectives and conclusions.

The next step of the process involved analyzing these statistics to determine important characteristics of the data. To perform this analysis, a couple methods were employed to get an understanding of the relationships with accuracy and in particular test accuracy. This included correlation, beta, coefficients from linear regression, and feature importance from random forest. Random forest produced the clearest results with the statistics most valuable and least valuable. These calculations were represented graphically through heatmaps and scatter plots.

The last part of this study was to find out what hyperparameters in human control that affect the statistics. The goal of this was to tune the hyperparameters that could improve the statistics and indirectly improve performance. Model summaries were stored as text files when the models initially ran. These summaries were scraped along with model names to pull the necessary information. The calculations described in the previous section (correlation, beta, linear regression coefficients, and random forest) were then applied on the hyperparameter data to predict the statistics.

Once these three methods were complete, we now had a method of running models and storing data, a method to find out statistics in the data that can be used to indirectly improve models, and a way to figure out what can be changed to influence these statistics.

Some hyperparameters remained constant through the project including 30 epochs to train, Adam and RMSprop optimizers, accuracy as the primary evaluation metric for performance, and the ReLU activation function for hidden layers.

**Results**

The final model used to predict the CIFAR-10 classes was a seven layer network with four convolutional layers and three dense fully connected layers utilizing dropout, l2_regularization, and batch normalization to achieve test accuracy performance of 74.7%, training accuracy of 96.3%, and validation accuracy of 75.6%. This model was constructed based on the findings of the underlying drivers of what moves neural networks. Broad themes that drove the methods to construct the models was the importance of maintaining weight variance, activation variance, activation, and maximum values low. The first few layers were paramount to the final performance of the model and it was crucial to regularize the higher layers because those values are sensitive and can have larger variances. Dense layers should be limited to only a few layers and avoid very high and low numbers of neurons. There should be a modest amount of CNN layers but not too much as the sweet spot was around three to four layers. A high amount of neurons should be used to represent the data. Extreme values should be avoided as there was a fine line between achieving an optimal model and a model tumbling over.

The first step of the analysis was finding the statistics that were most important in performance. Aggregated statistics alone achieved R squared scores of 81.2% with linear regression and 90.8% with random forest to predict test performance. These numbers were much higher than expected as I anticipated noise and not a simple relationship between performance and these statistics. Test accuracy was the focus as that is generally the north start when obtaining the best neural network model. Relationships most prevalent were the inverse relationships between weight variance, activation variance, and activation max. It was not surprising that these ended up being some of the more important stats. The four methods of calculating relationships (correlation, beta, linear regression coefficients, and random forest feature

importance) reinforced the graphical scatter plot with concrete numbers. While the conviction was not very high with these calculations, it was relatively clear that across the four, these statistics were prevalent. Correlation highlighted activation variance, activation max, and number of parameters with an inverse relationship, beta showed the same relationships, linear regression showed activation max, activation sparse neurons, and activation variance having the strongest negative relationship, and feature importance showing weight variance, activation variance, and activation max as being the most important. Training data performance showed similar relationships with the exception of a positive association between number of parameters and performance which is not very surprising. Less important statistics were the sparity of activations, amount of neurons below a threshold (although this was more helpful than the others), gradient variance, and gradient max values. These relationships can be seen in figures 1 and 2 in the appendix.

Applying these calculations layer-wise showed some important findings. This was approached with two methods, a similar systematic calculation like the evaluations and a graphical way by observing the changes in statistics through each epoch. The graphical method proved to be a more helpful way of determining the most useful statistics. Systematically, the easiest way of generalizing the results was by looking at a heatmap of all of the statistics where each was separated across the x-axis and the number of layers in the y-axis. The clear conclusion is that the lower layers have a much more influence on the results than the higher layers.

Utilizing the graph showed a few important observations that drove decision making. There was a clear disparity between the way better models traversed through time and poor models. Better models represented less volatile, low values, and decreasing acceleration of slopes whereas poor models showed the opposite. The difference was fairly obvious with girations much lower, values in particular weights and gradients less than half poor models, and slopes decelerating.

Poor performing models often had one or two layers going parabolic with some statistics. One of the most useful discoveries was the tendency for models to accelerate at an increasing pace through each epoch. Models representing this tendency had less parameters doing the predictions. Another interesting observation of a comparison between better and poorer models was the shape of the distribution of their statistics. Often, I would use this as a quick method of determining the need to change something in a model. Better models exhibited more narrow distributions in their statistics whereas poorer models exhibited wider distributions.

The last part of this analysis involved figuring out what can be done by a human to influence the statistics and indirectly improving performance. The method employed to carry this out was using a few key hyperparameters and model architectures. A similar calculation approach using the four methods (correlation, beta, linear regression coefficients, and random forest feature importance) were utilized. This was applied to the important statistics to find what drives those values. Overall findings showed that there should be a modest amount of CNN layers, only two to three dense fully connected layers, a higher amount of CNN neurons and not too many dense neurons. Better models exhibited ladder approaches of neurons and filter magnitudes. As you go higher into the network, increasing the amount of filters helped while after flattening and changing to dense fully connected layers, a decreasing laddered approach resulted in better performance. The most important normalization technique was dropout and l2 regularization. This analysis reiterated the importance of regularization but also showed where they were most useful. Applying each to obtain an optimal model required careful and not excessive tuning. Applying more than one of these techniques per layer often resulted in worse performance. A solution for a problem of higher layers substantially increasing in magnitude is to apply regularization techniques in particular l2 or batch normalization. Historically I have used batch normalization as a common way of standardizing data which made sense intuitively. However,

through this study, I have learned of the pitfall of overusing it and at the same time the importance of applying it in higher layers. These relationships can be seen through figure 3 in the appendix.

One of the more fascinating findings from this study was the inconsistent linear relationship between hyperparameters and model performance. This pitfall could be one of the more significant conclusions because individuals focused solely on training and validation accuracy would be subject to these challenges. Often simple heuristics are used to construct neural networks - if there is a gap between validation and training accuracy, reduce complexity and hyperparameters, if there isn't a gap, apply more complexity and hyperparameters. This would have resulted in a non-optimal model with this dataset and reiterates the benefit of using a more holistic approach utilizing parameter data. To achieve the desired parameter statistics, this often involved me increasing the complexity of a model to improve validation performance and not the contrast.

**Programming and Implementation**

Programming and implementation can be broken down into a couple pieces - importing, exploring, and preparing the data, creating the infrastructure to run the neural network, developing the calculation and graphs for the statistics, discovering the benefit of each statistic, and determining how to change the statistics through hyperparameters and layers.

Data was imported through the built in dataset into tensorflow. The data was explored to determine nuances including null values, frequency of classes, and maximum and minimum values. Plots were used to see what the data looked like to understand the complexity and noise. Data was prepared by standardizing, reshaping, and splitting out into train, validation, and split sets. Finally the data was put into slices to make it simple to obtain batches.

The second phase involved building out model architectures, using the gradient tape approach and running the model. Components of gradient tape involved specifying loss functions and optimizers for each split of data. Running the model was probably the longest function because it required all of the data to be initialized and collected while running the neural network. Performance was printed after each epoch to evaluate results.

After models were run, evaluation calculations were created including correlation, beta, linear regression, and random forest. These methods were developed separately and joined together along with graphing techniques to make implementation simpler. Results were displayed through heatmaps and scatter plots with seaborn.

Lastly, model summaries were scraped to grab number of layers, number of neurons and filters, and regularizations. These values were aggregated and then displayed through the same techniques in the previous section.

**Conclusion**

Overall the results of this analysis were encouraging showing the benefits of using statistical calculations of parameters to inform the tuning of neural networks. Broad trends and benefits were the importance of keeping weight and activation variance low, activation maximum values low, not too many parameters, focus on the beginning layers, and use sufficient regularization for higher layers. These generalizations helped develop the model with test accuracy performance of 74.7%, training accuracy of 96.3%, and validation accuracy of 75.6% on the CIFAR-10 dataset with CNNs. Neural networks predict complex relationships through latent space representations. Understanding these representations through simple performance numbers lacks the holistic approach of discovering underlying factors that drive results in a model. This framework to optimize a model can be approached in a systematic and

non-systematic way. Future studies will help determine the generalization across other datasets. Having this infrastructure in the place will make it easier to apply these techniques to other projects.

**Appendix**

Figure 1 - Relationship between statistics and test accuracy

Figure 1a


Figure 1b

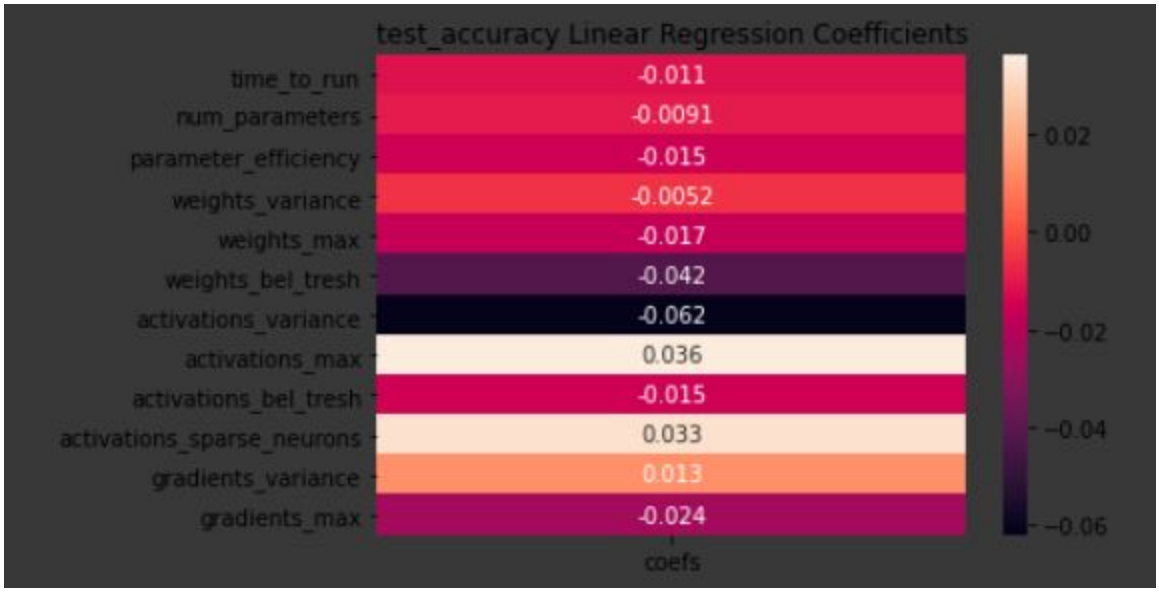

test_accuracy Linear Regression Coefficients
time_to_run
num_parameters
parameter_efficiency
weights_variance
weights_max
weights_bel_tresh
activations_variance
activations_max
activations_bel_tresh
activations_sparse_neurons
gradients_variance
gradients_max
-0.011
-0.0091
-0.015
-0.0052
-0.017
-0.042
-0.062
0.036
-0.015
0.033
0.013
-0.024
coefs
0.02
0.00
−0.02
−0.04
−0.06

Figure 1c

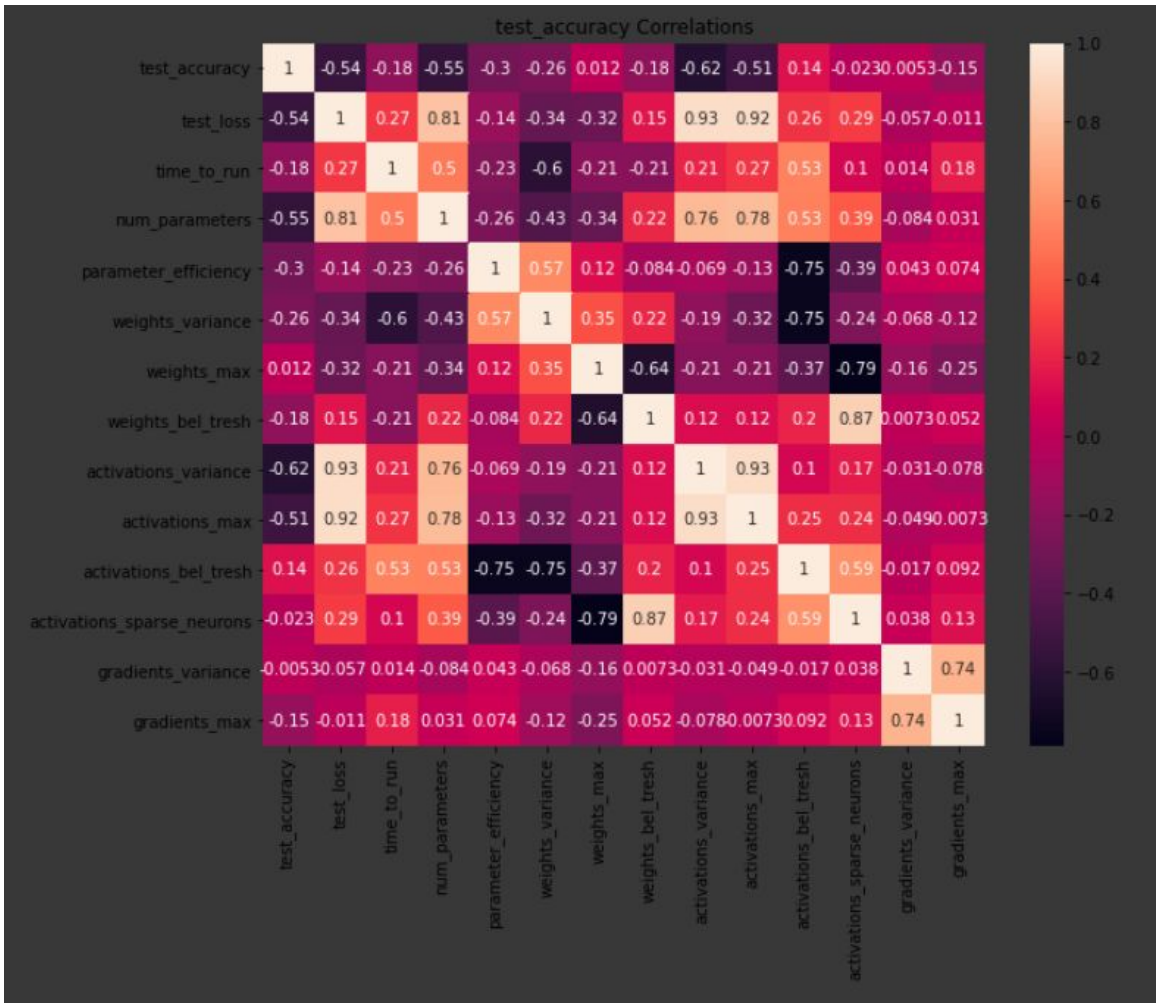



Figure 1d

Figure 2a. - Final model summary statistics through time

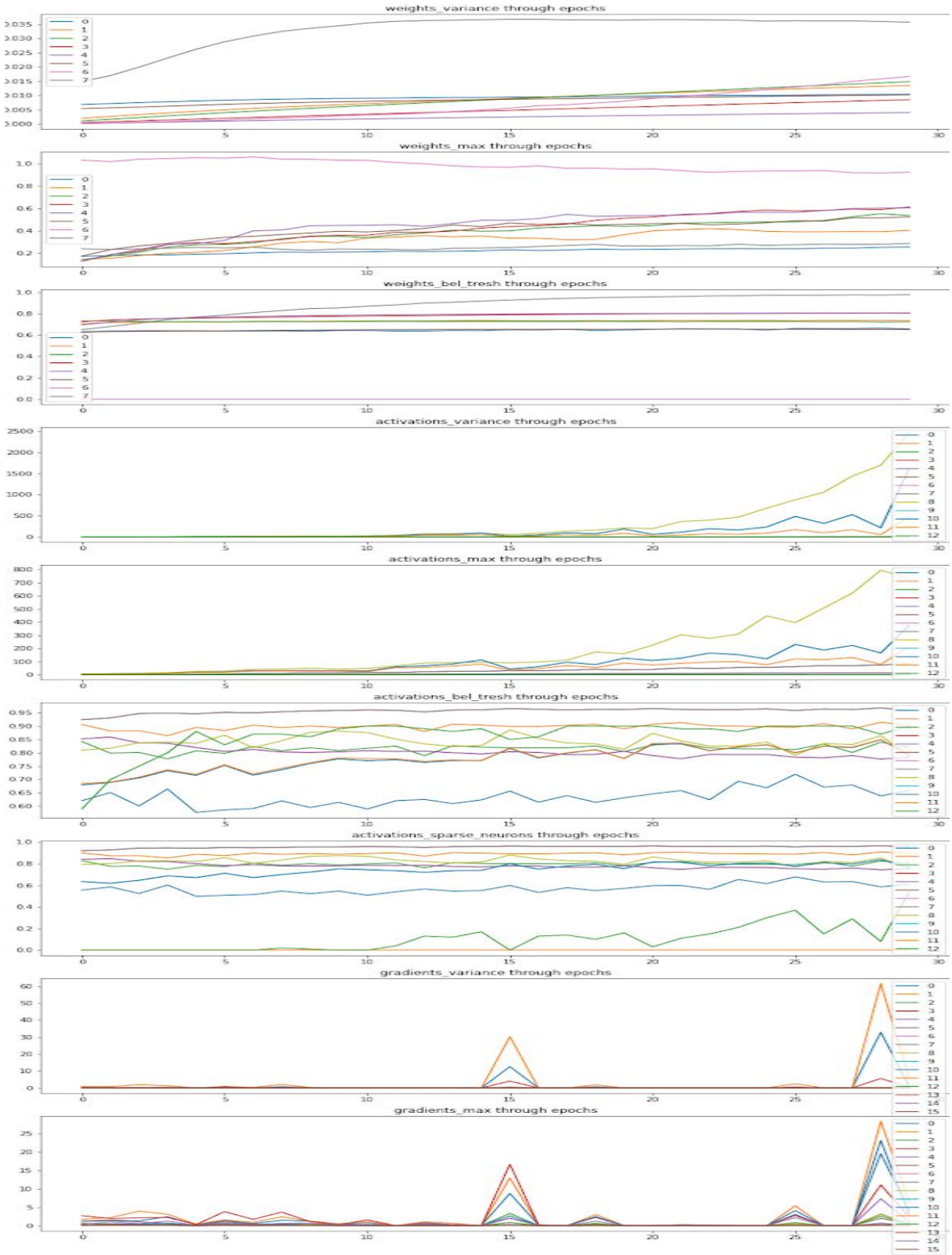



Figure 2b. Final model layer-wise graph of statistics

max', 'weights_bel_tresh', 'activations_variance', 'activations_max', 'activations_bel_tresh', 'activations_sparse_neurons

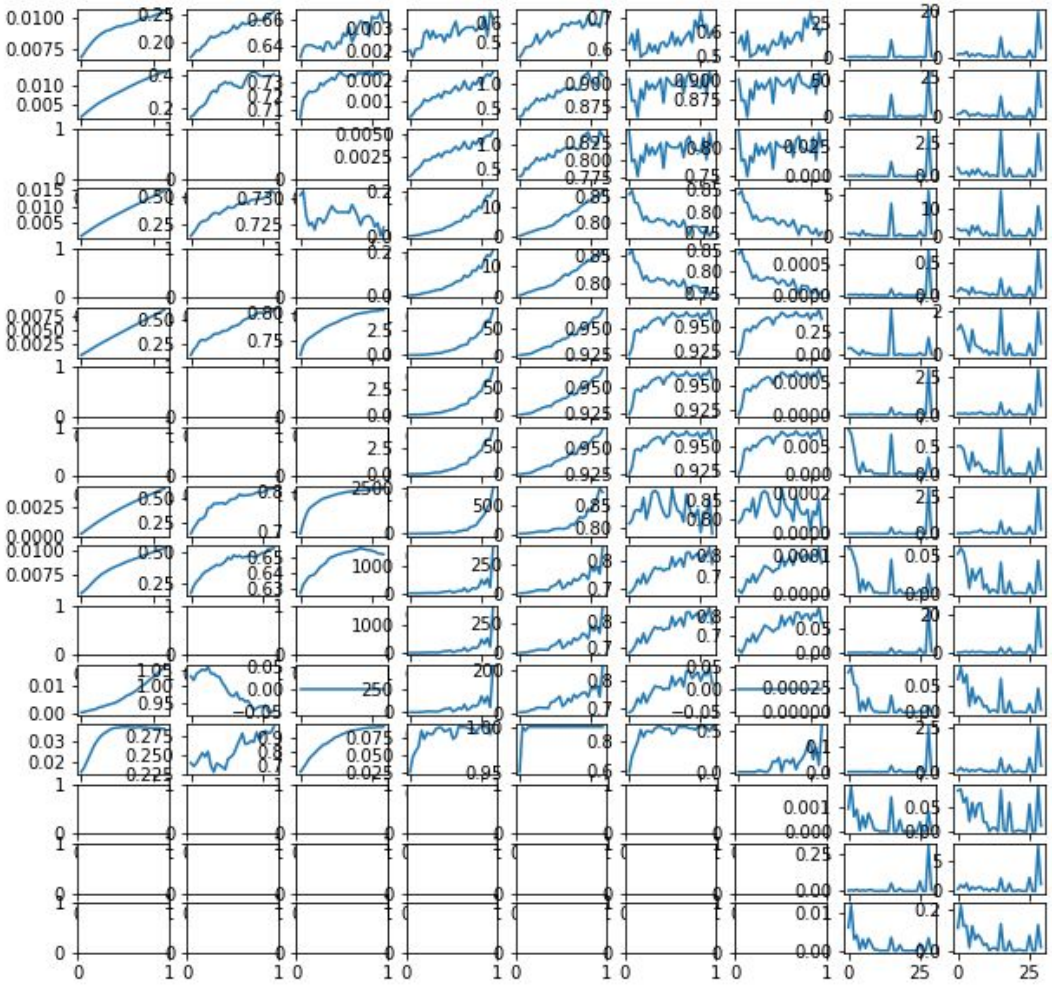


Figure 2c. Final model statistical distributions per layer

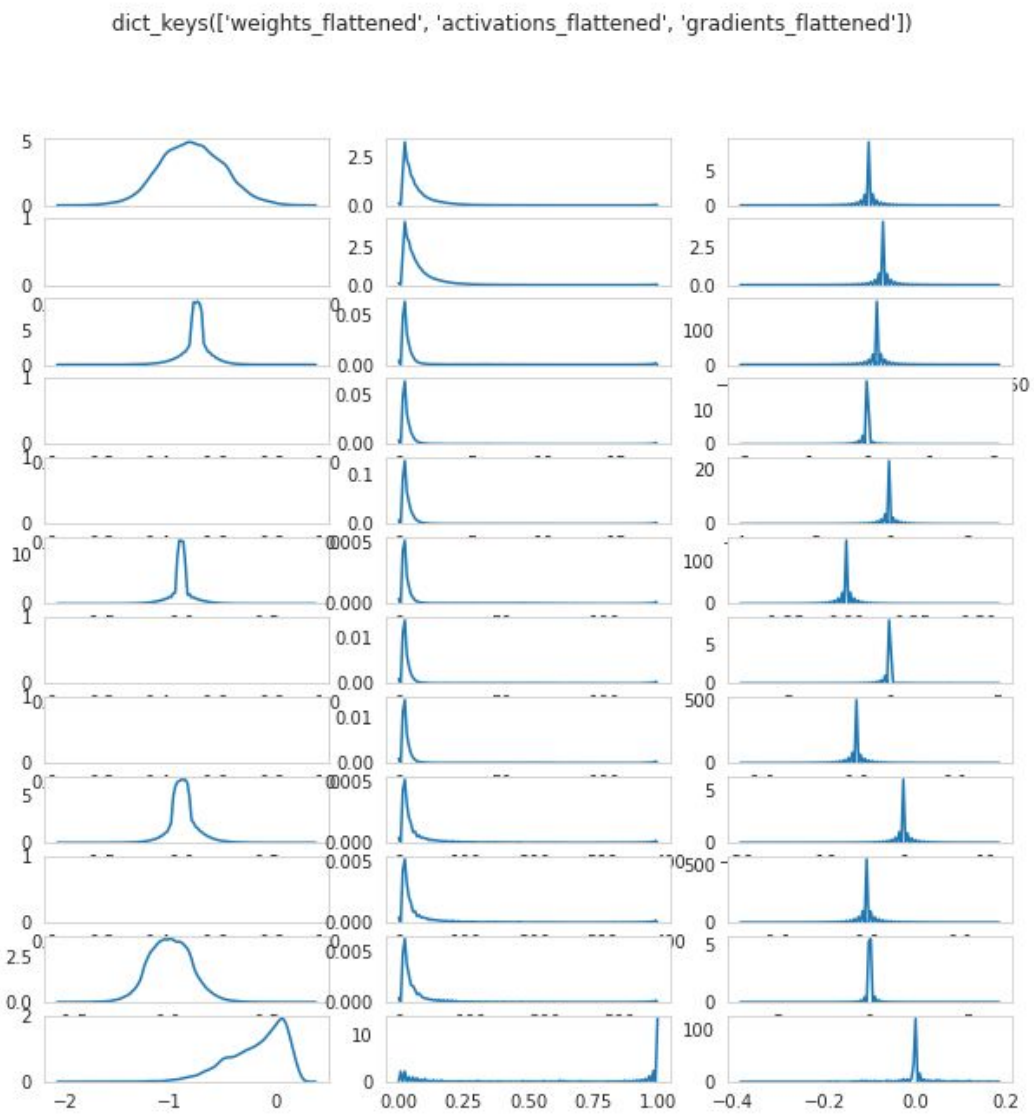



Figure 3 - Hyperparameter and layer tuning for activation variance statistic
Figure 3a - Random Forest feature importance


Figure 3b - Linear regression coefficients

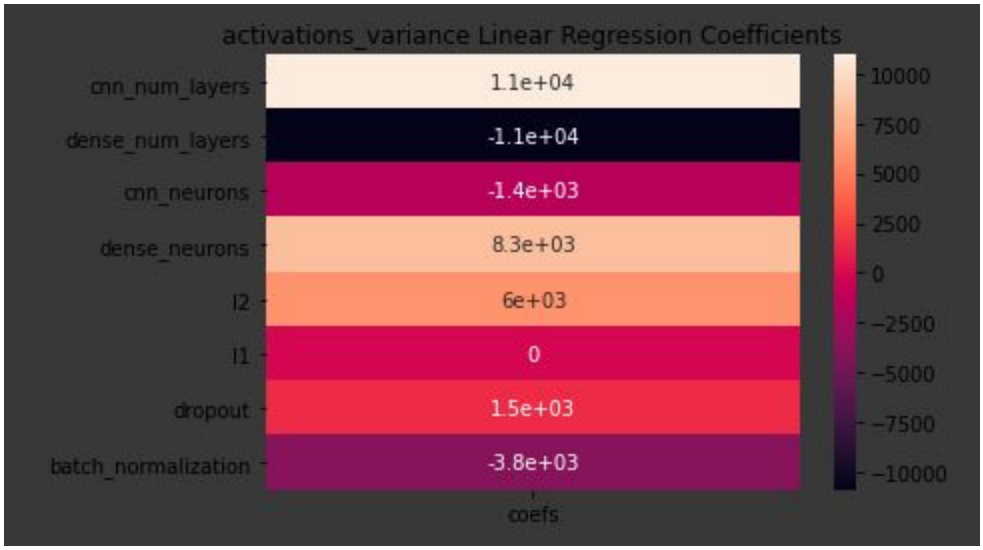
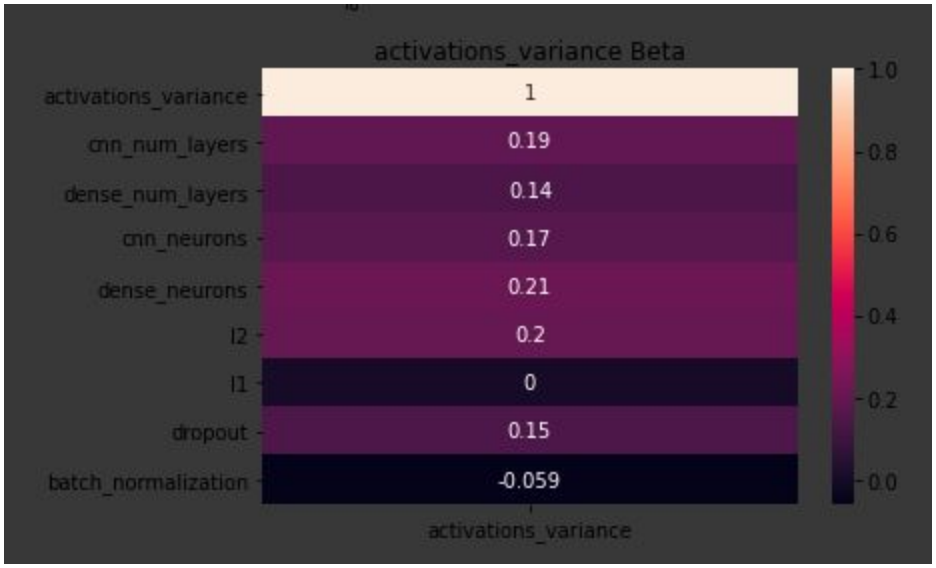



Figure 3c - Correlations


Figure 3d - Beta



**References**


Krizhevsky, A., Sutskever, I., and Hinton. G. (2012). ImageNet classification with deep convolutional neural networks. NIPS, Vol.1, 1097-1105. 10.5555

Krizhevsky, A. (2009, April 8). Learning Multiple Layers of Features from Tiny Images. Retrieved June 13, 2020, from https://www.researchgate.net/publication/265748773_Learning_Multiple_Layers_of_Features_from_Tiny_Images

Bengio, Y. (2012, January 01). Practical recommendations for gradient-based training of deep architectures. Retrieved June 13, 2020, from

Bengio, Y. (2012, January 01). Practical recommendations for gradient-based training of deep architectures. Retrieved June 13, 2020, from https://link.springer.com/chapter/10.1007/978-3-642-35289-8_26https://link.springer.com/chapter/10.1007/978-3-642-35289-8_26

Bergstra, J., Yamins D., and Cox, D.D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proc. of ICML, pages 115–123, 2013.

LeCun, Y., Bottou, L., Bengio Y., and Haffner P. (1998). Gradient-Based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278-2324.

Optimization of Machine Learning Hyperparameters on Large Datasets. Retrieved June 13, 2020, from https://arxiv.org/abs/1605.07079

Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional neural networks.

arXiv preprint arXiv:1311.2901, 2013.