Rajiv Jarhad
2022UG000034
ADVST

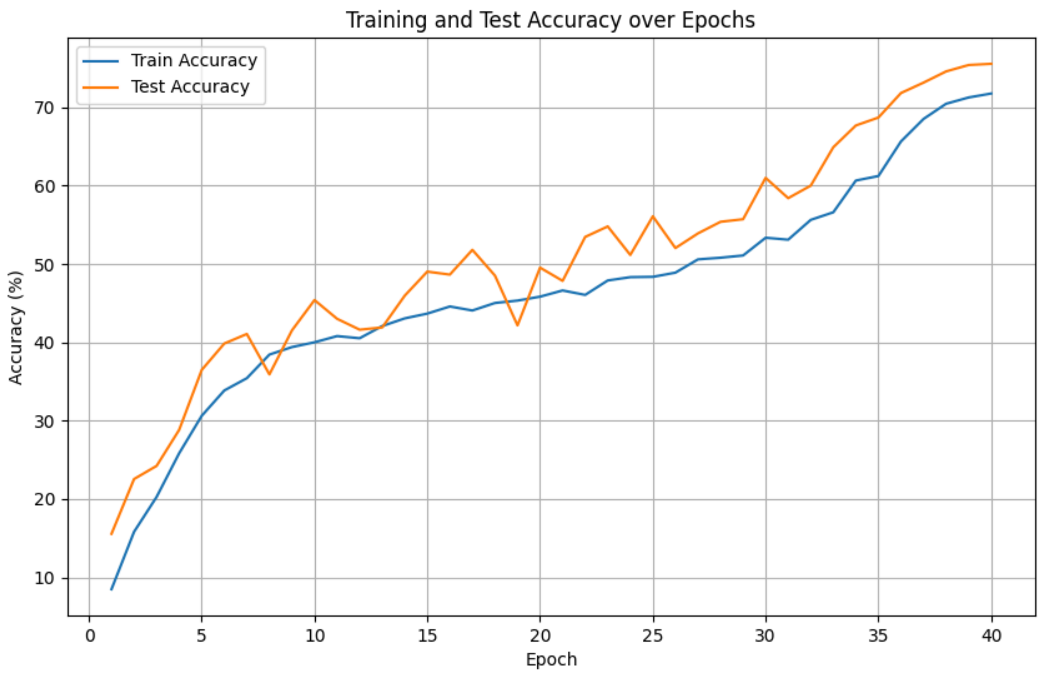
Question 1: Weight/Bias Evolution and Empirical Spectral Density

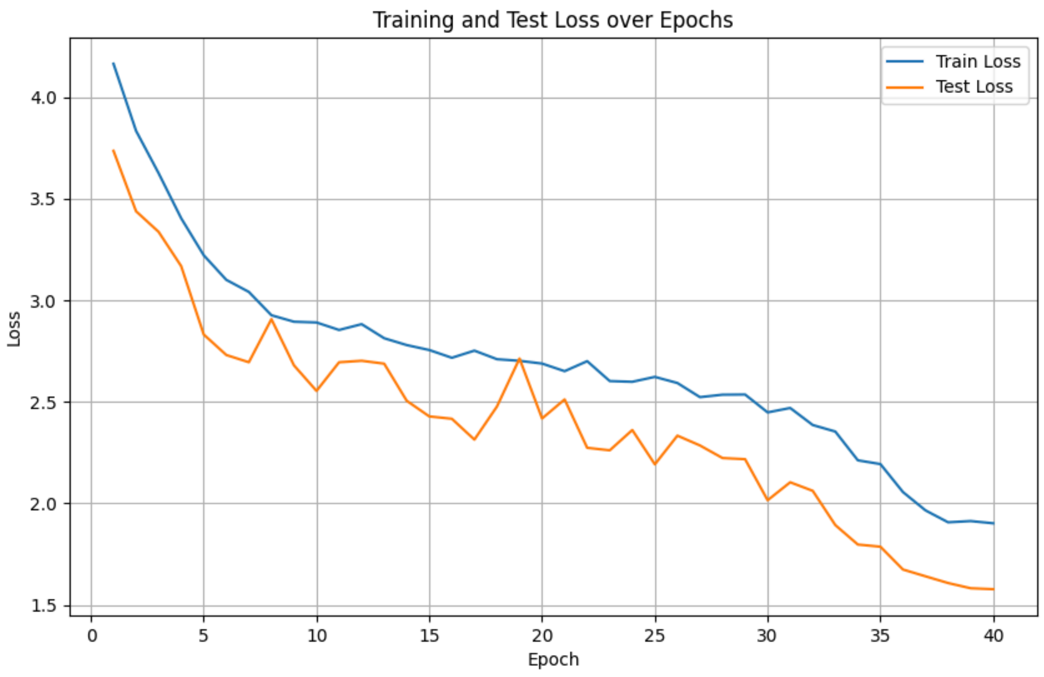
This project analyses the training behaviour of a neural network on the CIFAR-100 dataset by visualising the evolution of weights and biases over time. Using normalised histograms and Empirical Spectral Density (ESD) plots, it captures how different layers of the network respond across epochs. Power-law exponents from the ESDs offer insights into the model's generalisation capabilities. The visualisations are enriched with animations and interactive elements to provide a clear and engaging understanding of the learning process.

The initial phase of this assignment involved training a Convolutional Neural Network (CNN) model, where I experimented with various architectures, including a self-designed structure and several combinations to optimise performance. After a thorough comparison of the results, I determined that the ResNet-18 architecture, enhanced with additional techniques such as mixup data augmentation and a OneCycleLR scheduler, delivered the best accuracy in a minimal number of epochs. The training process utilised a comprehensive Python script that leveraged libraries like PyTorch, torchvision, and Matplotlib, setting up directories for plots, histograms, and model checkpoints to track progress. The code incorporated advanced data preprocessing with transforms like RandomCrop, RandomHorizontalFlip, and ColorJitter, alongside normalisation tailored to CIFAR-100's statistical properties. Furthermore, the implementation featured a custom ResNet-18 model with BasicBlock layers, optimised using SGD with momentum and weight decay, and included histogram plotting for each epoch to visualise weight and bias distributions, providing deeper insights into the learning dynamics over the 40 epochs of training.

Throughout the training process, I generated histograms to visualise the evolution of weights and biases, capturing their distributions from the initialised model (designated as epoch 0) through to the final model across all epochs. These histograms, saved alongside the stored model checkpoints, provided a clear view of how the parameters adjusted over time. Additionally, I plotted accuracy and loss curves to monitor performance, observing an accuracy exceeding 75%

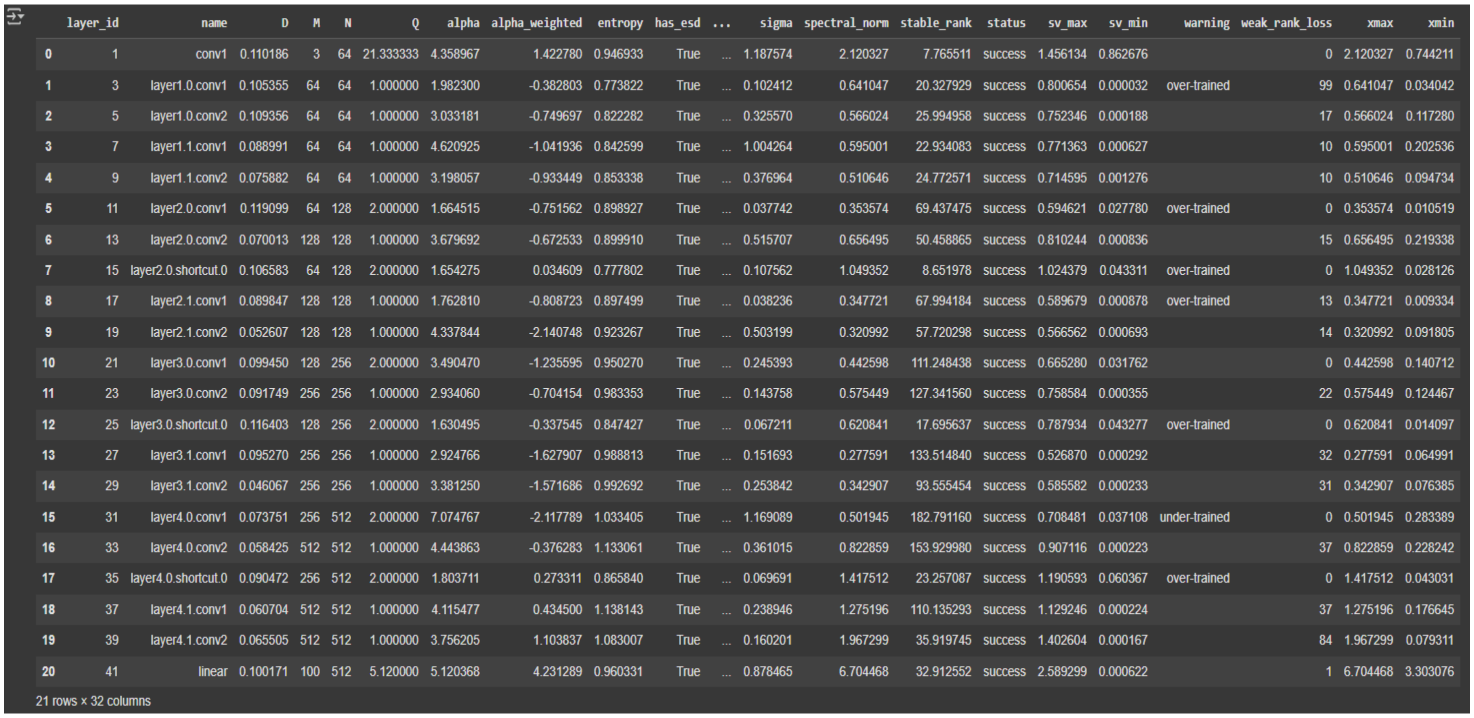
accompanied by a consistent decrease in loss, indicating a stable and effective training progression that aligned well with the learning objectives for the CIFAR-100 dataset.





After completing the initial training phase, I shifted my focus to evaluating the WeightWatcher library to analyse Empirical Spectral Density (ESD) and Alpha values, which provided deeper insights into the neural network's behaviour on the CIFAR-100 dataset. Concurrently, I experimented with other Python-based tools like Manim and Matplotlib to develop enhanced video visualisations, encountering numerous trials and errors as I refined the process. Ultimately, after extensive testing, I discovered that integrating Plotly offered the most effective results, delivering superior video outputs for plotting the evolution of weights and biases with improved clarity and interactivity.

The details which Weight Watchers gave:

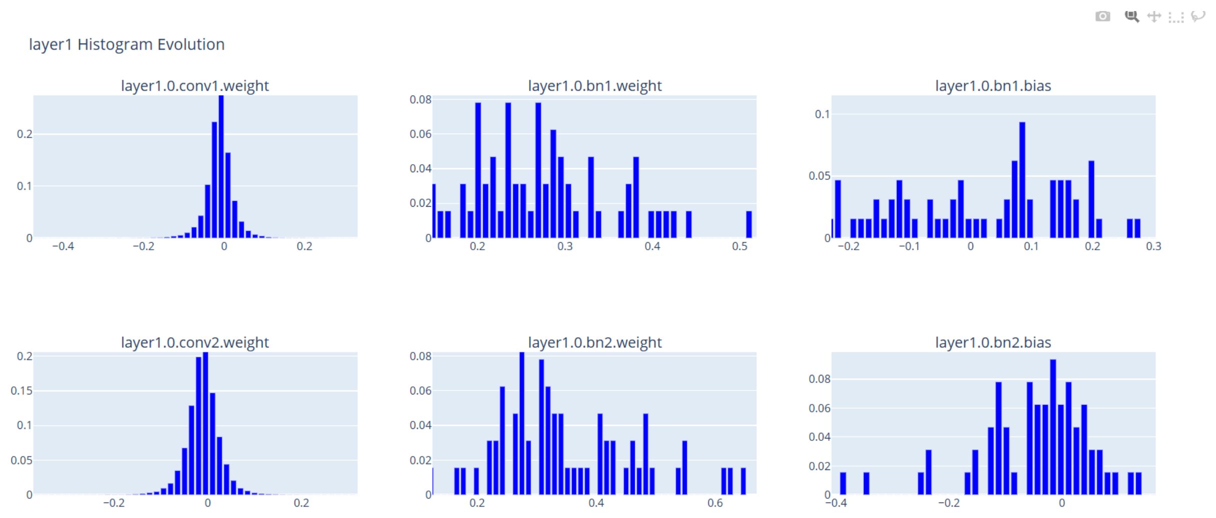


	layer_id	name	D	M	N	Q	alpha	alpha_weighted	entropy	has_esd	...	sigma	spectral_norm	stable_rank	status	sv_max	sv_min	warning	weak_rank_loss	xmax	xmin
0	1	conv1	0.110186	3	64	21.333333	4.358967	1.422780	0.946933	True	...	1.187574	2.120327	7.765511	success	1.456134	0.862676		0	2.120327	0.744211
1	3	layer1.0.conv1	0.105355	64	64	1.000000	1.982300	-0.382803	0.773822	True	...	0.102412	0.641047	20.327929	success	0.800654	0.000032	over-trained	99	0.641047	0.034042
2	5	layer1.0.conv2	0.109356	64	64	1.000000	3.033181	-0.749697	0.822282	True	...	0.325570	0.566024	25.994958	success	0.752346	0.000188		17	0.566024	0.117280
3	7	layer1.1.conv1	0.088991	64	64	1.000000	4.620925	-1.041936	0.842599	True	...	1.004264	0.595001	22.934083	success	0.771363	0.000627		10	0.595001	0.202536
4	9	layer1.1.conv2	0.075882	64	64	1.000000	3.198057	-0.933449	0.853338	True	...	0.376964	0.510646	24.772571	success	0.714595	0.001276		10	0.510646	0.094734
5	11	layer2.0.conv1	0.119099	64	128	2.000000	1.664515	-0.751562	0.898927	True	...	0.037742	0.353574	69.437475	success	0.594621	0.027780	over-trained	0	0.353574	0.010519
6	13	layer2.0.conv2	0.070013	128	128	1.000000	3.679692	-0.672533	0.899910	True	...	0.515707	0.656495	50.458865	success	0.810244	0.000836		15	0.656495	0.219338
7	15	layer2.0.shortcut.0	0.106583	64	128	2.000000	1.654275	0.034609	0.777802	True	...	0.107562	1.049352	8.651978	success	1.024379	0.043311	over-trained	0	1.049352	0.028126
8	17	layer2.1.conv1	0.089847	128	128	1.000000	1.762810	-0.808723	0.897499	True	...	0.038236	0.347721	67.994184	success	0.589679	0.000878	over-trained	13	0.347721	0.009334
9	19	layer2.1.conv2	0.052607	128	128	1.000000	4.337844	-2.140748	0.923267	True	...	0.503199	0.320992	57.720298	success	0.566562	0.000693		14	0.320992	0.091805
10	21	layer3.0.conv1	0.099450	128	256	2.000000	3.490470	-1.235595	0.950270	True	...	0.245393	0.442598	111.248438	success	0.665280	0.031762		0	0.442598	0.140712
11	23	layer3.0.conv2	0.091749	256	256	1.000000	2.934060	-0.704154	0.983353	True	...	0.143758	0.575449	127.341560	success	0.758584	0.000355		22	0.575449	0.124467
12	25	layer3.0.shortcut.0	0.116403	128	256	2.000000	1.630495	-0.337545	0.847427	True	...	0.067211	0.620841	17.695637	success	0.787934	0.043277	over-trained	0	0.620841	0.014097
13	27	layer3.1.conv1	0.095270	256	256	1.000000	2.924766	-1.627907	0.988813	True	...	0.151693	0.277591	133.514840	success	0.526870	0.000292		32	0.277591	0.064991
14	29	layer3.1.conv2	0.046067	256	256	1.000000	3.381250	-1.571686	0.992692	True	...	0.253842	0.342907	93.555454	success	0.585582	0.000233		31	0.342907	0.076385
15	31	layer4.0.conv1	0.073751	256	512	2.000000	7.074767	-2.117789	1.033405	True	...	1.169089	0.501945	182.791160	success	0.708481	0.037108	under-trained	0	0.501945	0.283389
16	33	layer4.0.conv2	0.058425	512	512	1.000000	4.443863	-0.376283	1.133061	True	...	0.361015	0.822859	153.929980	success	0.907116	0.000223		37	0.822859	0.228242
17	35	layer4.0.shortcut.0	0.090472	256	512	2.000000	1.803711	0.273311	0.865840	True	...	0.069691	1.417512	23.257087	success	1.190593	0.060367	over-trained	0	1.417512	0.043031
18	37	layer4.1.conv1	0.060704	512	512	1.000000	4.115477	0.434500	1.138143	True	...	0.238946	1.275196	110.135293	success	1.129246	0.000224		37	1.275196	0.176645
19	39	layer4.1.conv2	0.065505	512	512	1.000000	3.756205	1.103837	1.083007	True	...	0.160201	1.967299	35.919745	success	1.402604	0.000167		84	1.967299	0.079311
20	41	linear	0.100171	100	512	5.120000	5.120368	4.231289	0.960331	True	...	0.878465	6.704468	32.912552	success	2.589299	0.000622		1	6.704468	3.303076

21 rows × 32 columns

In my analysis, I employed Plotly to create interactive visualisations, which proved to be a pivotal tool for exploring the weights and biases of the neural network trained on the CIFAR-100 dataset. This choice followed extensive experimentation with various Python libraries, where Plotly stood out for its ability to generate dynamic and engaging video outputs that enhanced the interpretability of parameter evolution across epochs. The interactive nature of these plots allowed for a more detailed and user-friendly examination of the data, building on the successful outcomes observed during previous trials with this library.

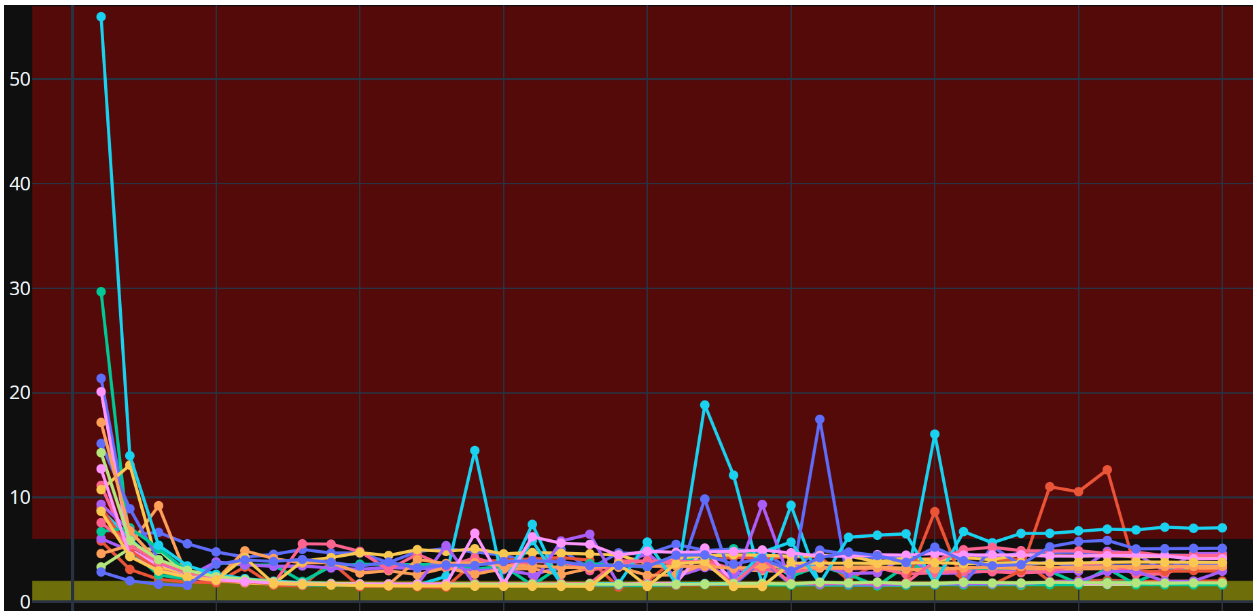
The interface looks like this:



The most challenging task I tackled was calculating the Empirical Spectral Density (ESD), saving Alpha values, and generating log-log plots, requiring an intensive coding approach to handle the complexity of the analysis. I implemented a loop using the WeightWatcher library, which efficiently computed and stored Alpha values for each layer and epoch, streamlining the process and laying a solid foundation for subsequent analysis. This methodical approach, building on our earlier discussions about optimising ResNet-18 for CIFAR-100, significantly simplified my workflow. To further enhance the visualisation, I created a video compilation of all the log-log plots, leveraging tools like Plotly to dynamically showcase the spectral evolution across the training process.

The initial phase of this assignment focused on training a Convolutional Neural Network (CNN) model, where I experimented with various architectures, including a custom design and combinations, ultimately finding that ResNet-18, enhanced with mixup and a OneCycleLR scheduler, delivered the best accuracy in fewer epochs. The Python script, leveraging PyTorch and Matplotlib, established directories for plots and checkpoints, implemented advanced transforms such as RandomCrop and ColorJitter, and included a custom ResNet-18 with histogram plotting to monitor weight and bias changes over 40 epochs. With this phase completed, I moved into the final stage of extracting Alpha values and plotting them, utilising the weightwatcher library’s prior calculations to create interactive visualisations with Plotly.

After extracting the Alpha values, I developed an interactive visualisation using Plotly, enhancing the analysis of the CIFAR-100 dataset by creating a dynamic and user-friendly representation of the spectral trends across epochs and layers.



The interactive visualisation of Alpha values, developed using Plotly for the CIFAR-100 dataset analysis, revealed promising results, with the majority of values falling within the expected range of 2 to 6, as recommended for optimal model performance. To enhance interpretability, I colour-coded the Alpha values, using yellow to highlight those below 2, which indicate overfitting, and reserved colours for values above 6 to signify underfitting. This colour scheme, as seen in the Plotly chart, effectively distinguished the spectral trends across epochs and layers, confirming the robustness of the training process with the ResNet-18 model.

Through this assignment, I deepened my understanding of neural network dynamics and the critical factors that drive performance. **Model architecture** proved essential in shaping how effectively a network learns, while **optimisation techniques** were key to refining its accuracy. I also learnt the importance of **monitoring model health**, which revealed patterns like underfitting or overfitting, guiding adjustments throughout the process. The assignment underscored the value of **iterative refinement**, showing how continuous tweaks based on observations can elevate results. Applying these insights to a complex dataset like CIFAR-100 highlighted their real-world relevance, solidifying my grasp of foundational machine learning principles.

- To Experience the Visualisations: https://rajiv714.github.io/ADVST/
- To Access the Complete Assignment Folder: https://drive.google.com/drive/folder
- To Access the Python Code File: https://drive.google.com/file
- To Access all the Saved Models: https://drive.google.com/drive/folders
- To Access Data Related to Weights and Bias:https://drive.google.com/drive
- To Access Data of Alpha: https://drive.google.com/drive/folders/
- To Access ESD LogLog Plots: https://drive.google.com/drive/folders