# WeightWatcher: A Diagnostic Tool for Deep Neural Networks

Charles H. Martin PhD, Calculation Consulting

(in joint with Michael Mahoney, UC Berkeley)

pip install weightwatcher

# Open source tool: weightwatcher

https://github.com/CalculatedContent/WeightWatcher

**WeightWatcher (WW)**: is an open-source, diagnostic tool for analyzing Deep Neural Networks (DNN), without needing access to training or even test data. It can be used to:

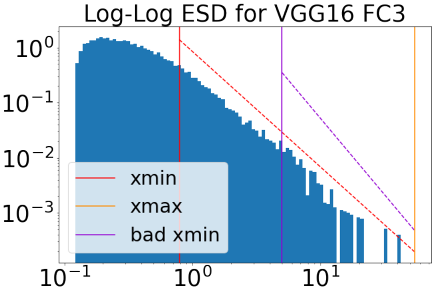
- analyze pre/trained pyTorch and keras models
- inspect models that are difficult to train
- gauge improvements in model performance
- predict test accuracies across different models
- detect potential problems when compressing or fine-tuning pretrained models

It is based on theoretical research (done injoint with UC Berkeley) into *Why Deep Learning Works*, using ideas from Random Matrix Theory (RMT), Statistical Mechanics, and Strongly Correlated Systems.

**pip install weightwatcher**

# Shape and Scale Metrics

**WeightWatcher (WW)**: analyzes the shape and scale of the correlations in the layer weight matrices:




**WW**: extracts, plots, and fits the Empirical Spectral Density (ESD, or eigenvalues) for each layer weight matrix (or tensor slice).

The *tail of the ESD* contains the most informative components.

The shape of the tail carries useful information!

# WeightWatcher: Usage

**Usage**

import weightwatcher as ww
watcher = ww.WeightWatcher(model=model)
details = watcher.analyze(plot=True)
summary = watcher.get_summary(details)

| | layer_id | name | D | M | N | alpha | alpha_weighted | has_esd | lambda_max | layer_type | ... | rand_num_spikes | rand_sigma_mp | ra |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | None | 0.240111 | 3.0 | 64.0 | 2.400712 | 2.627967 | 1.0 | 12.435451 | LAYER_TYPE.CONV2D | ... | 135.0 | 1.000000 | |
| **1** | 5 | None | 0.112669 | 64.0 | 128.0 | 7.116304 | 4.721276 | 1.0 | 4.607285 | LAYER_TYPE.CONV2D | ... | 25.0 | 0.551250 | |
| **2** | 8 | None | 0.076209 | 128.0 | 256.0 | 2.981087 | 1.739893 | 1.0 | 3.833927 | LAYER_TYPE.CONV2D | ... | 17.0 | 0.451562 | |
| **3** | 10 | None | 0.068890 | 256.0 | 256.0 | 5.667264 | 2.600458 | 1.0 | 2.876445 | LAYER_TYPE.CONV2D | ... | 0.0 | 0.935068 | |
| **4** | 13 | None | 0.084938 | 256.0 | 512.0 | 2.593428 | 1.432684 | 1.0 | 3.568032 | LAYER_TYPE.CONV2D | ... | 8.0 | 0.431523 | |
| **5** | 15 | None | 0.038416 | 512.0 | 512.0 | 3.309962 | 2.216486 | 1.0 | 4.673487 | LAYER_TYPE.CONV2D | ... | 0.0 | 0.939111 | |
| **6** | 18 | None | 0.052924 | 512.0 | 512.0 | 3.446656 | 1.859810 | 1.0 | 3.464163 | LAYER_TYPE.CONV2D | ... | 0.0 | 0.888574 | |
| **7** | 20 | None | 0.034290 | 512.0 | 512.0 | 3.261262 | 2.524426 | 1.0 | 5.943799 | LAYER_TYPE.CONV2D | ... | 0.0 | 0.942012 | |
| **8** | 25 | None | 0.032563 | 4096.0 | 25088.0 | 2.325065 | 3.583809 | 1.0 | 34.784030 | LAYER_TYPE.DENSE | ... | 1.0 | 0.898506 | |
| **9** | 28 | None | 0.030891 | 4096.0 | 4096.0 | 2.167513 | 3.858526 | 1.0 | 60.278519 | LAYER_TYPE.DENSE | ... | 1.0 | 0.959863 | |
| **10** | 31 | None | 0.039773 | 1000.0 | 4096.0 | 2.825653 | 4.999373 | 1.0 | 58.786867 | LAYER_TYPE.DENSE | ... | 206.0 | 1.000000 | |

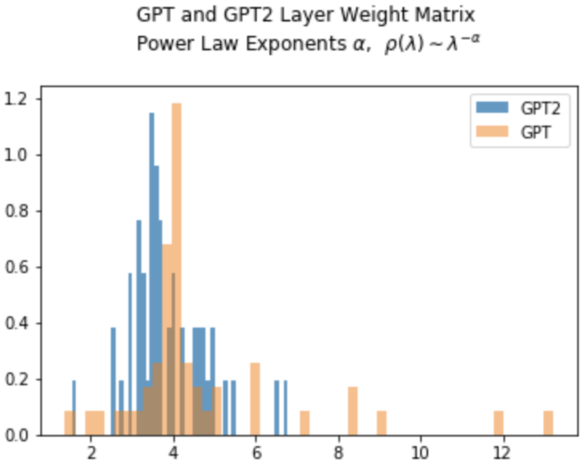
summary = watcher.get_summary(details)

```
{'log_norm': 2.11,
 'alpha': 3.06,
 'alpha_weighted': 2.78,
 'log_alpha_norm': 3.21,
 'log_spectral_norm': 0.89,
 'stable_rank': 20.90,
 'mp_softrank': 0.52}]
```

# Layer-by-Layer Analysis

**WW layer metrics**: can detect potential problems *in the ESD shapes*

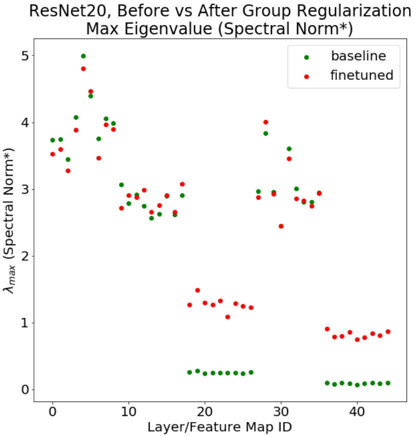
Poorly trained models (orange) can have unusually large layer $\alpha$'s.



# Layer-by-Layer Analysis

**WW layer metrics**: can detect potential problems *in the ESD Scales*

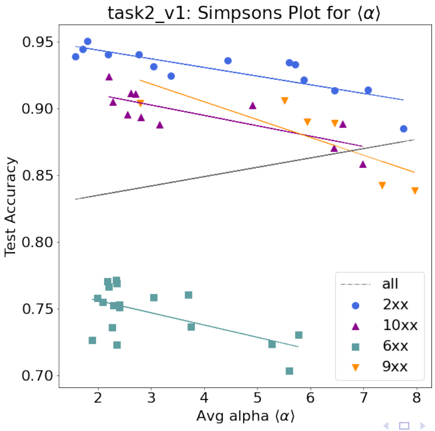
Compressed models (red) can show unexpected scale changes




example from Intel distiller Group Regularization technique

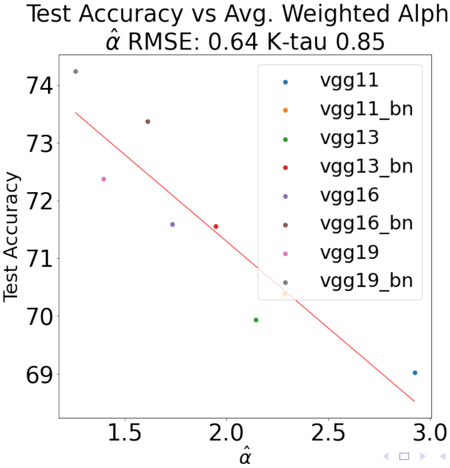
# $\alpha$: a regularization metric

**The WW $\langle\alpha\rangle$ metric**: predicts test accuracy for a given model (i.e same depth) when varying the regularization hyper-parameters (such as batch size, weight decay, momentum, etc.)—*without access to the test or training data.*



# $\hat{\alpha}$: a multi-purpose metric

**The WW $\hat{\alpha}$ metric**: predicts test accuracy for models in the same architecture series across varying depth and other architecture parameters and regularization hyper-parameters—-*without access to the test or training data.*



# Research Update: Early Stopping with $\alpha$

**Early Stopping**: Our HTSR theory suggests that when our PL metric $\alpha < 2.0$, the layer may be overtrained. Moreover, the early stopping should be applied when the layer averaged $\langle\alpha\rangle < 2.0$.


Early results by Xander Dunn show this amazingly well!
Here, when training a transformer model, as the training error decreases, $\alpha$ decreases. But just when the test error starts to increase, $\alpha < 2.0$.
This is remarkable, and we are pursuing this on a wide variety of models.

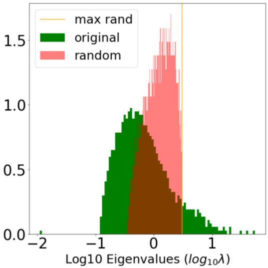
# Research Update: Early Stopping with $\alpha$

**Early Stopping**: Our HTSR theory suggests that when our PL metric $\alpha < 2.0$, the layer may be overtrained. Moreover, the early stopping should be applied when the layer averaged $\langle\alpha\rangle < 2.0$.

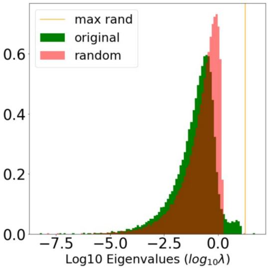

Early results by Xander Dunn show this amazingly well!
Here, when training a transformer model, as the training error decreases, $\alpha$ decreases. But just when the test error starts to increase, $\alpha < 2.0$.
This is remarkable, and we are pursuing this on a wide variety of models.

# Research Update: Heavy Tails in **W** and **X**

**Correlation Traps** We *conjecture* that when the unusually large elements $W_{i,j}$ arise in the weight matrices, these act like traps (in the ESD of **X** that prevent good generalization.




(a) ESD of **W** and randomized **W**.




(b) ESD of **W** and randomized **W**.

These can be seen using the [randomize] option in **weightwatcher**.
We beleive we can use this to detect overfitting and are investigating this.