Notes on the paper “Flat Channels to Infinity in Neural Loss Landscapes” by Flavio Martinelli

Compiled by D. Gueorguiev, 9/1/25

Notation:

Capital bold italic letter denotes a matrix e.g.

Small cap bold italic letter a denotes a column vector e.g.

represents element-wise multiplication

# Introductory notes

In the evolving field of deep learning, researchers continue to uncover hidden dynamics in the training of neural networks that challenge traditional assumptions; the study shows that seemingly flat valleys in neural loss landscapes may hide “wormhole-like channels” where neurons’ weights diverge toward infinity, leading them to fuse into gated linear units (GLUs) and demonstrating that even simple fully connected layers can spontaneously develop gating mechanisms; this suggests that optimization is not just about finding minima but can induce emergent functional shifts, with potential business impacts ranging from unexpected model capabilities to new cost-performance tradeoffs in AI-driven products, though the effect remains mostly theoretical and observed in limited regression settings, meaning enterprises should view it as an intriguing signal of emergent design rather than a replacement for deliberate architectural innovation.

The loss landscapes of neural networks contain minima and saddle points that may be connected in flat regions or appear in isolation. The authors of this paper identify and characterize a special structure in the loss landscape: channels along which the loss decreases extremely slowly, while the output weights of at least two neurons,  and , diverge to , and their input weight vectors,  and , become equal to each other. At convergence, the two neurons implement a gated linear unit:

Geometrically, these channels to infinity are asymptotically parallel to symmetry-induced lines of critical points. Gradient flow solvers, and related optimization methods like SGD or ADAM, reach the channels with high probability in diverse regression settings, but without careful inspection they look like flat local minima with finite parameter values. The paper characterizes these quasi-flat regions in terms of gradient dynamics, geometry, and gives functional interpretation. The emergence of gated linear units at the end of the channels highlights a surprising aspect of the computational capabilities of fully connected layers.

A diagram of a graph

AI-generated content may be incorrect.

# References

[1] [Flat Channels to Infinity in Neural Loss Landscapes, Flavio Martinelli et al, EPFL, 2025](https://github.com/dimitarpg13/deep_learning_and_neural_networks/blob/main/literature/articles/Flat_Channels_to_Infinity_in_Neural_Loss_Landscapes_Martinelli_2025.pdf)

[2] [Convex Optimization, Steven Boyd, Lieven Vandenberghe, 2009](https://github.com/dimitarpg13/optimization_classification_regression/blob/main/literature/books/ConvexOptimization_Boyd_2004.pdf)

[3] [Stochastic Gradient Descent, Convex Optimization 10-725, CMU, slides, Ryan Tibshirani](https://github.com/dimitarpg13/statistical_learning_and_kernel_methods/blob/main/literature/articles/gradient_descent/stochastic-gradient-descent_Ryan_Tibshirani_slides.pdf)

[4] [Gating Mechanism, Wikipedia](https://en.wikipedia.org/wiki/Gating_mechanism)

[5] [Gated Recurrent Unit, Wikipedia](https://en.wikipedia.org/wiki/Gated_recurrent_unit)

[6] [Understanding LSTM: a Tutorial into Long Short-Term Memory Recurrent Neural Networks, Ralf C. Staudemeyer, Eric Rothstein Morris, 2019](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/TutorialOnLongShortTermMemory2019.pdf)

[7] [Long Short-Term Memory, Sepp Hochreiter, Juergen Schmidhuber, Neural Computation 9(8), 1997](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/LongShortTermMemory.pdf)

# Appendix

Notation:

Capital bold italic letter denotes a matrix e.g.

Small cap bold italic letter a denotes a column vector e.g. ; the corresponding row vector is denoted as

represents element-wise multiplication between two vectors or two matrices of equal sizes e.g.

denotes a column-wise concatenation of column vectors and

denotes a column-wise concatenation of matrices and of equal widths

denotes a row-wise concatenation of row vectors and

denotes a row-wise concatenation of matrices and of equal heights

## Gating mechanism in Long Short-Term Memory (LSTM)

The primary reason for the gating mechanism introduced in LSTM is to mitigate the vanishing gradient problem which is observed with the vanilla RNNs.

LSTM unit consists of 3 gates:

An *input gate*, which controls the flow of new information into the memory cell

A *forget gate*, which controls how much information is retained from the previous time step

An *output gate*, which controls how much information is passed to the next layer

The functioning of those gates is modeled by the following set of equations

(1)

(2)

(3)

(4)

(5)

(6)

denotes the *weight matrix* for the input gate, connecting the *current input* to the input gate activation function .

denotes the *weight matrix* for the previous *hidden state* , connecting the latter to the input gate activation function .

The dimension of the activation state vector is equal to the dimension of the hidden state vector:

The dimensions of and are determined by the size of the input and the number of hidden units in the LSTM layer. If the size of the input vector is and the number of hidden units represented by the hidden state vector is then the weight matrix for the input gate will have dimensions . Similarly, the weight matrix for the hidden state will have dimensions .

Equivalent notation to write equations (1) – (4) is:

(7)

(8)

(9)

(10)

Here

The role of the input gate

LSTMs use gates to control the flow of information, allowing them to selectively remember or forget data over long sequences. The input gate is responsible for deciding how much of the new information from the current input should be stored in the cell state. The gate’s activation vector involves a combination of the previous hidden state and the current input offset by the input bias .