Winning the Initialization Lottery:
Faster Training and Better Performance through Weight Initialization

Diego Aguirre, Ph.D.

Progress Report: 04/24/22

Team Motivation and Objectives

- The number of parameters in models is growing (almost) exponentially.
  - Need more compute, more data, more time, more money, etc.
- We want self-supervised models that:
  - Are not unnecessarily large
  - Perform well
  - Have a reasonable inference time

Team Efforts

We are tackling the problem from different angles.

Today's Presenters

Hao Tang	Yen Meng / Ray Chen	Diego Aguirre
Tiny SSL	Sequence Compression	Weight Initialization

The Importance of Weight Initialization

Weight Initialization: Why?

The Lottery Ticket Hypothesis

> "…dense, randomly-initialized, feed-forward networks contain subnetworks ('winning tickets') that - when trained in isolation - reach test accuracy comparable to the original network in a similar number of iterations. The winning tickets we find have won the initialization lottery: their connections have initial weights that make training particularly effective."

[Frankle, J., & Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635]

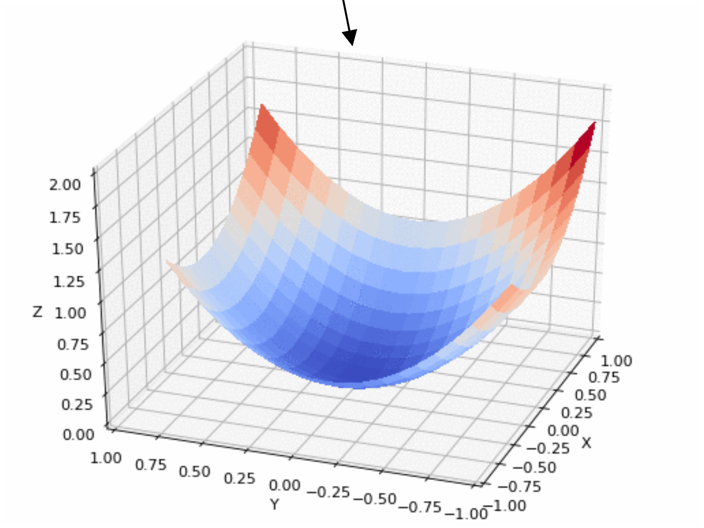
Weight Initialization: Why?

We know* that:

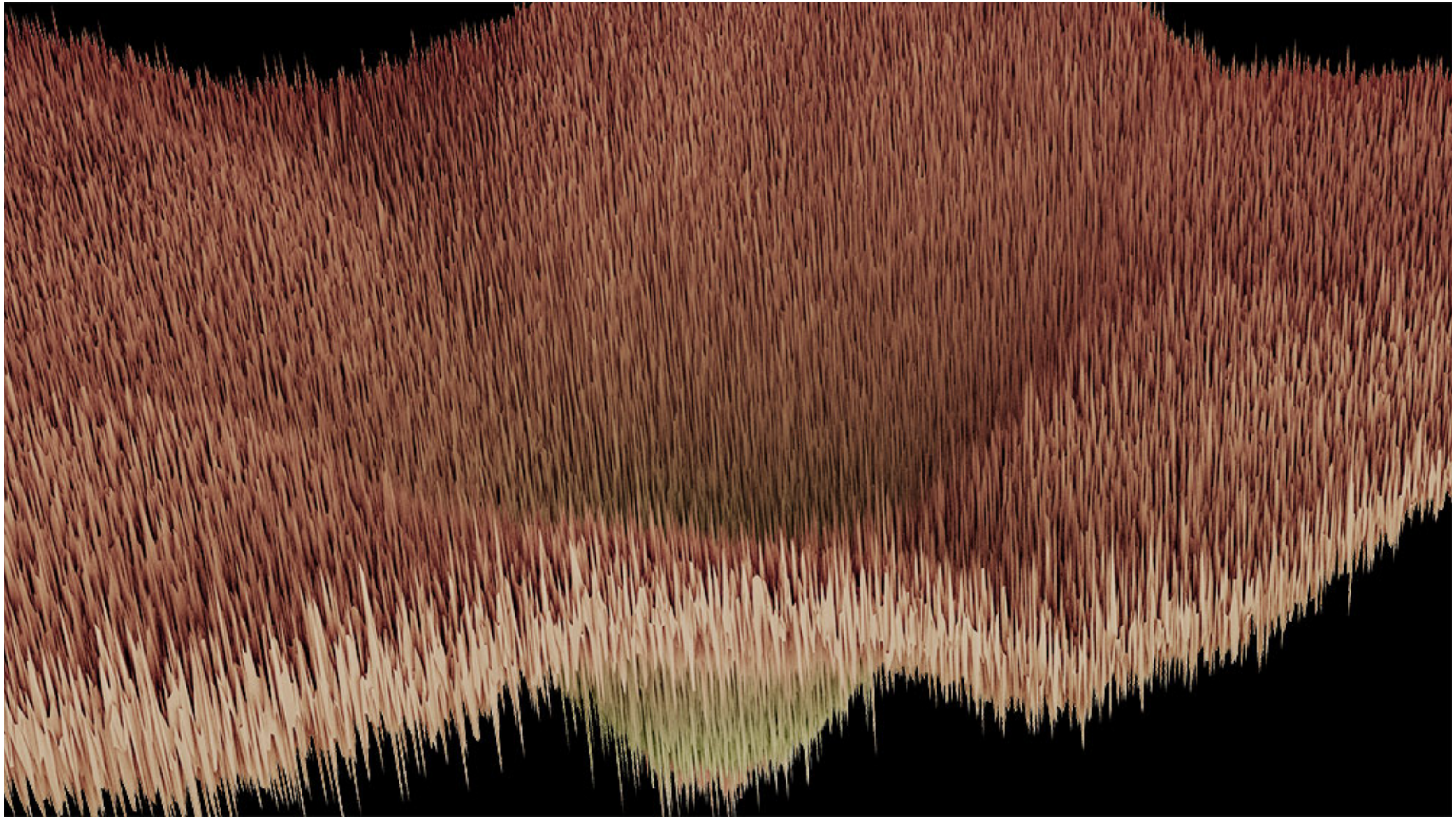
- Large models can be compressed
- “Lottery tickets” trained in isolation perform almost as well as the original network

Popular solution: buy as many tickets as you can afford

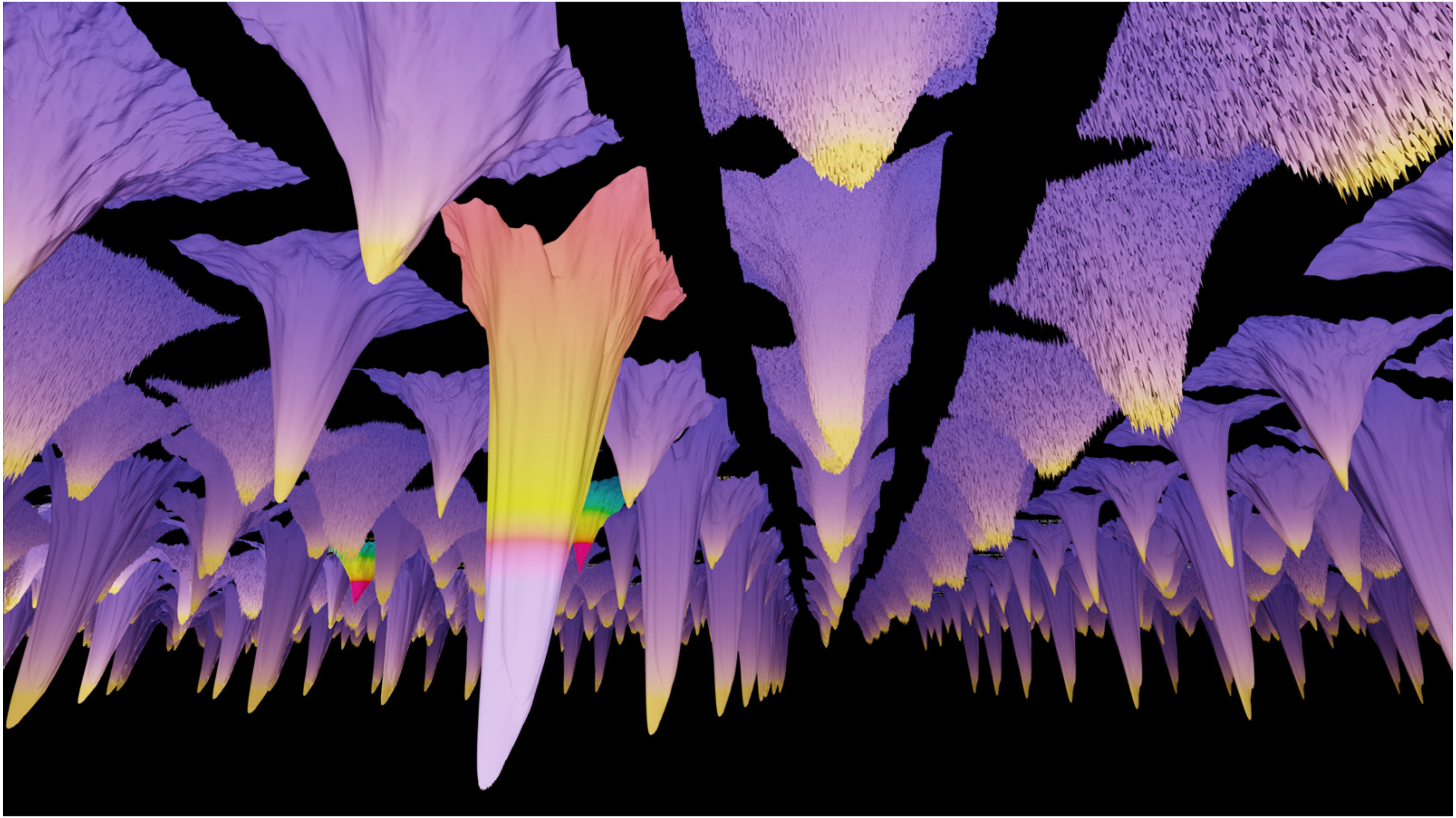
Ideal loss landscape



As networks become larger, the loss landscape is more likely to become *chaotic* and *highly non-convex*.



https://losslandscape.com/



https://losslandscape.com/

Weight Initialization: Questions

- What makes an initialization good? What properties should initial weight values have to increase the likeliness of finding a good minimum?
- How can we leverage training data and model architecture knowledge in the weight initialization process?
- To what extend can a good weight initialization strategy improve training time / model performance?
- Can a good weight initialization strategy alleviate the need for a large, trained network to build a smaller one? If so, to what extend?

Weight Initialization
HuBERT

Proposed Initialization

1. Initialize weight matrices of fully-connected layers using orthonormal matrices. Initialize kernels in convolutional layers to be orthonormal. Set biases to 0.

SVD decomposition / Gram–Schmidt process

Why? Speeds up convergence relative to the standard Gaussian initialization with iid weights

[Hu, W., Xiao, L., & Pennington, J. (2020). Provable benefit of orthogonal initialization in optimizing deep linear networks. *ICLR 2020*]

Proposed Initialization

2. Re-scale weights so the standard deviation of a layer's outputs is 1.

a. Select a subset of the training data to serve as the *initialization set* (k-means or some other criteria)

b. Feed initialization set to the network and compute a layer's output

c. W = W / std(output)

Why? Faster training and (in some cases) better performance

[Mishkin, D., & Matas, J. (2015). All you need is a good init. ICLR 2016]

Proposed Initialization

3. Initialize biases to control for initial non-linearity (act_prob hyperparam)

a. Select a subset of the training data to serve as the *initialization set* (k-means or some other criteria)

b. Feed initialization set to the network and compute a layer's output before ReLU / ReLU-like activation

c. To initialize a unit's bias, find activation value *v* from unit's outputs, such that the fraction of outputs > v = act_prob, and set bias term to *-v*

Why? Faster training and better performance

[Aguirre, D., & Fuentes, O. (2019, September). Improving weight initialization of relu and output layers. In International Conference on Artificial Neural Networks]

Proposed Initialization

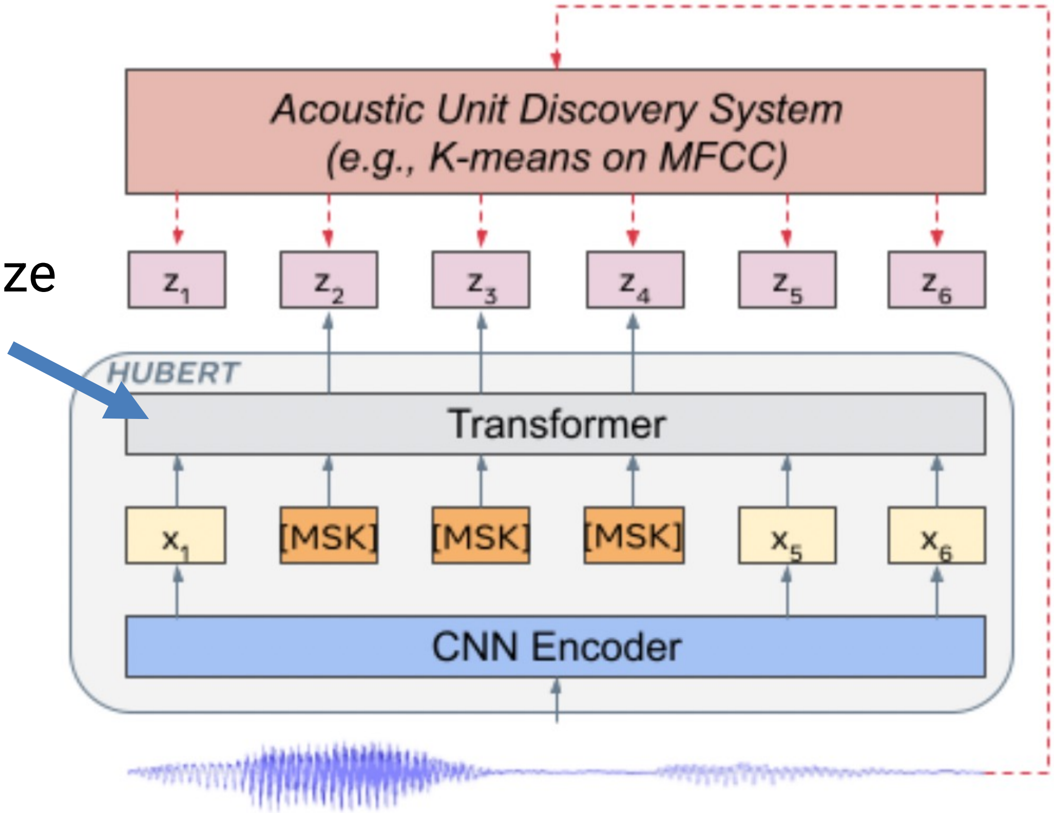
3. Initialize biases to control for initial non-linearity (act_prob hyperparam)

act_prob = 0.75

-3	-3 - (-2) = -1
-2	-2 - (-2) = 0
-1	-1 - (-2) = 1
3	3 - (-2) = 5
4	4 - (-2) = 6
5	5 - (-2) = 7
7	7 - (-2) = 9
9	9 - (-2) = 11

HuBERT

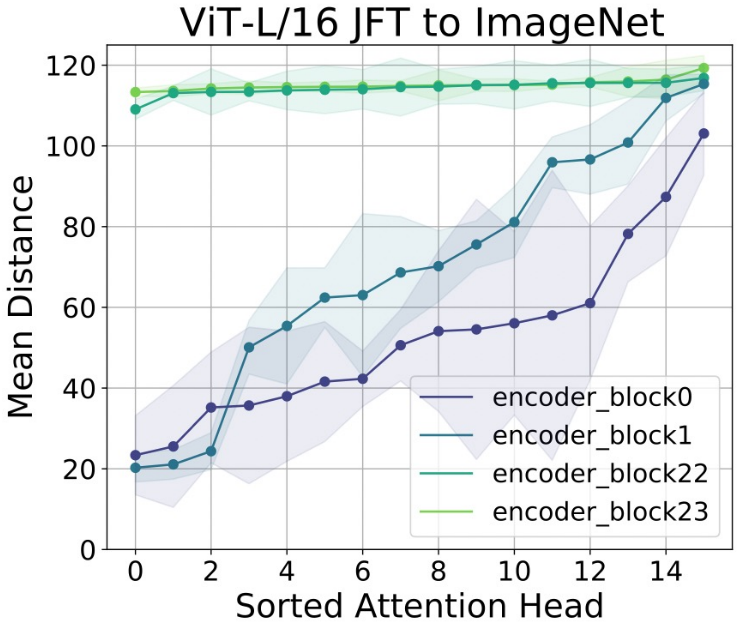
How should we initialize Transformers?



What Do Transformers Learn?

Attention heads in earlier layers attend both locally and globally

Attention heads in later layers attend globally



[Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., & Dosovitskiy, A. (2021). Do vision transformers see like convolutional neural networks?. Advances in Neural Information Processing System]

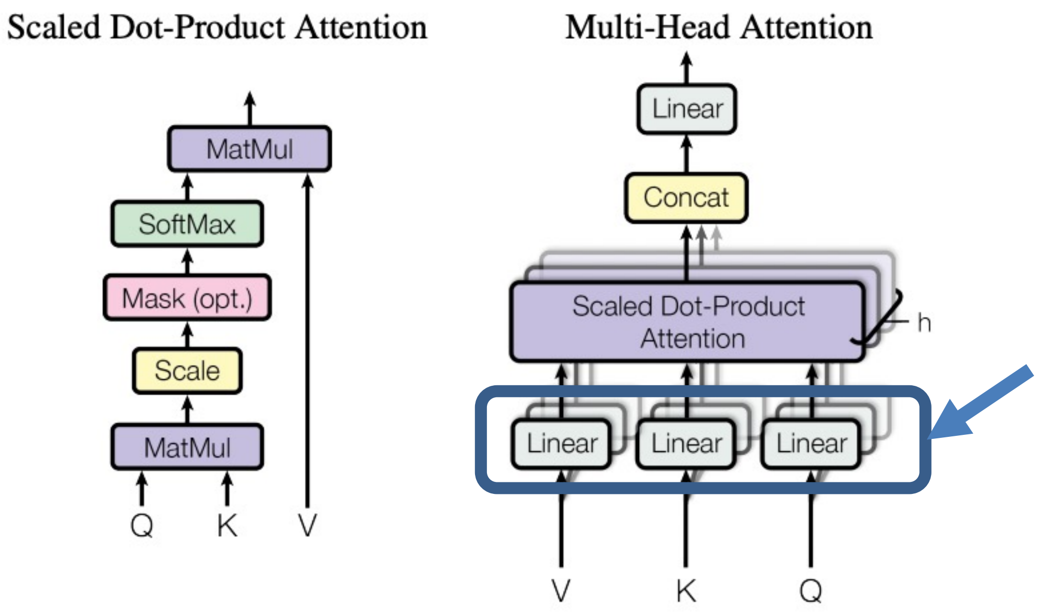
Pre-training On Artificial Data

- Pre-training language models on artificial data with specific traits and no semantics results in better downstream performance than training from scratch.

- Important traits: better performance is achieved when the tokens in the artificial dataset have longer range dependencies.

[Chiang, C. H., & Lee, H. Y. (2021). On the Transferability of Pre-trained Language Models: A Study from Artificial Datasets. arXiv preprint arXiv:2109.03537.]

Transformer Architecture



Use what we just discussed as intuition to initialize these

Proposed Initialization

4. Initialize Transformer

a. Initialize Wq, Wk, Wv matrices of all attention heads in an attention layer using orthonormal vectors

b. Compute desired attention scores for each attention head

Heads in earlier layers should attend locally and globally and heads in later layers should attend mostly globally

head_att_score_logits[i,j] = sample from normal distribution where mean and std are a function of i, j, head_id/num_heads, layer_id/num_layers

Proposed Initialization

c. Feed initialization set and solve for Wq (or Wk) using pseudo inverse / Linear SVM

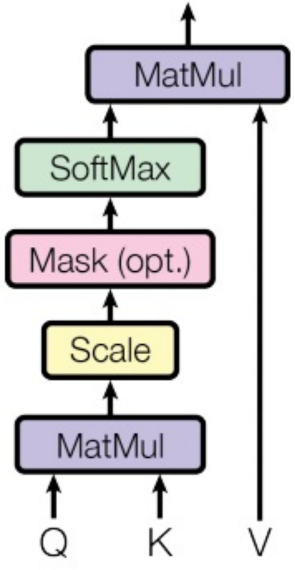
(Emb * Wq) * K.T /sqrt(dim) = logits of desired attention scores

(Emb * Wq) * K.T = logits of desired attention scores * sqrt(dim)

(Emb * Wq) * K.T = L

Emb * Wq = L * pseudo_inv(K.T)

Wq = pseudo_inv(Emb) * L * pseudo_inv(K.T)



Other Ideas

- GLU(a ,b) = a $\otimes$ σ(b)

  Initialize bias of layer that produces (a, b) to ensure that values in b are >= hyperparam

- Maintain desirable properties through regularization

- Look for properties / patterns in “lottery ticket winners” and use them to improve weight initialization strategy

Timeline

April – May

- Complete implementation
- Run initial experiments – compare default initialization vs proposed one

  Train for a limited number of steps on LibriSpeech – compare loss values

- Learn and repeat

June – July

- Train to completion – compare against publicly available pre-trained HuBERT

  Performance on SUPERB tasks

- Train smaller models – compare performance
- Integrate with team's innovations

Questions?

Thank you!

Diego Aguirre

daguirre6@utep.edu