Rupert Mitchell

Self-Expanding Neural Networks

Many currently popular machine learning methods can be summarized as "fit a large parameterized model to some data with gradient descent." This, however, elides much of the difficulty in choosing the parameterization. In this work we address the choice of size (e.g. width and depth) for a neural network by allowing this to automatically adapt during training. In particular, we want to avoid both making an arbitrary, often excessive, choice of size, and treating size as a hyperparameter which must be tuned over the course of multiple expensive restarts. It is our hope that this will result in fewer irrelevant features to interpret, and a more comprehensible story about how and why they got there.

Factorizing Network Expansion

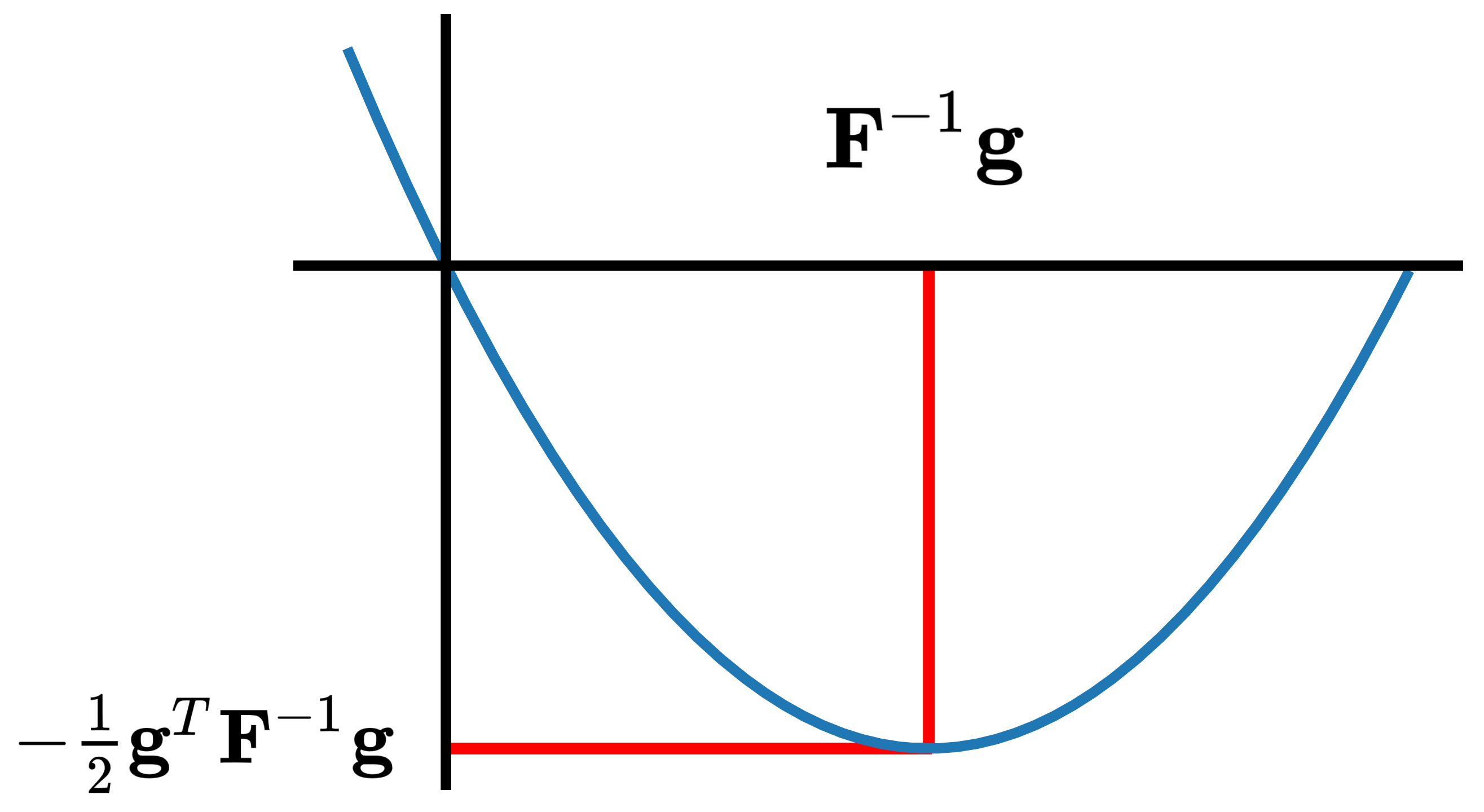
We can break the problem into three key questions:

- **When** should we add parameters?
- **Where** should we add them? Expand an existing layer or insert a new one?
- **What** should the initialisation degrees of freedom be set to?

If we have some notion of how useful any particular expansion of our parameterisation is we can answer all of these.

A Second Order Measure of Usefulness

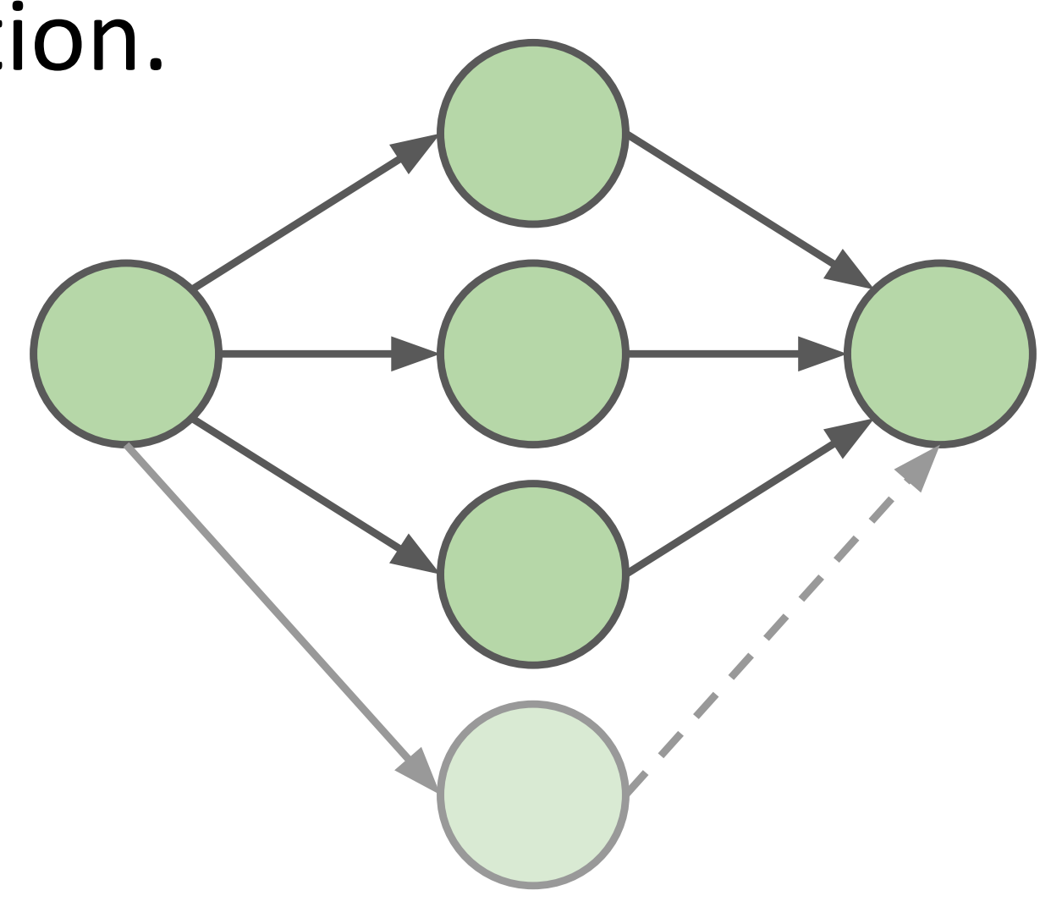
Given a second order approximation to our function we can estimate the improvement in loss at convergence. (Note that we denote the Fisher matrix, which is a curvature approximation, here by **F** and the gradient as **g**.)



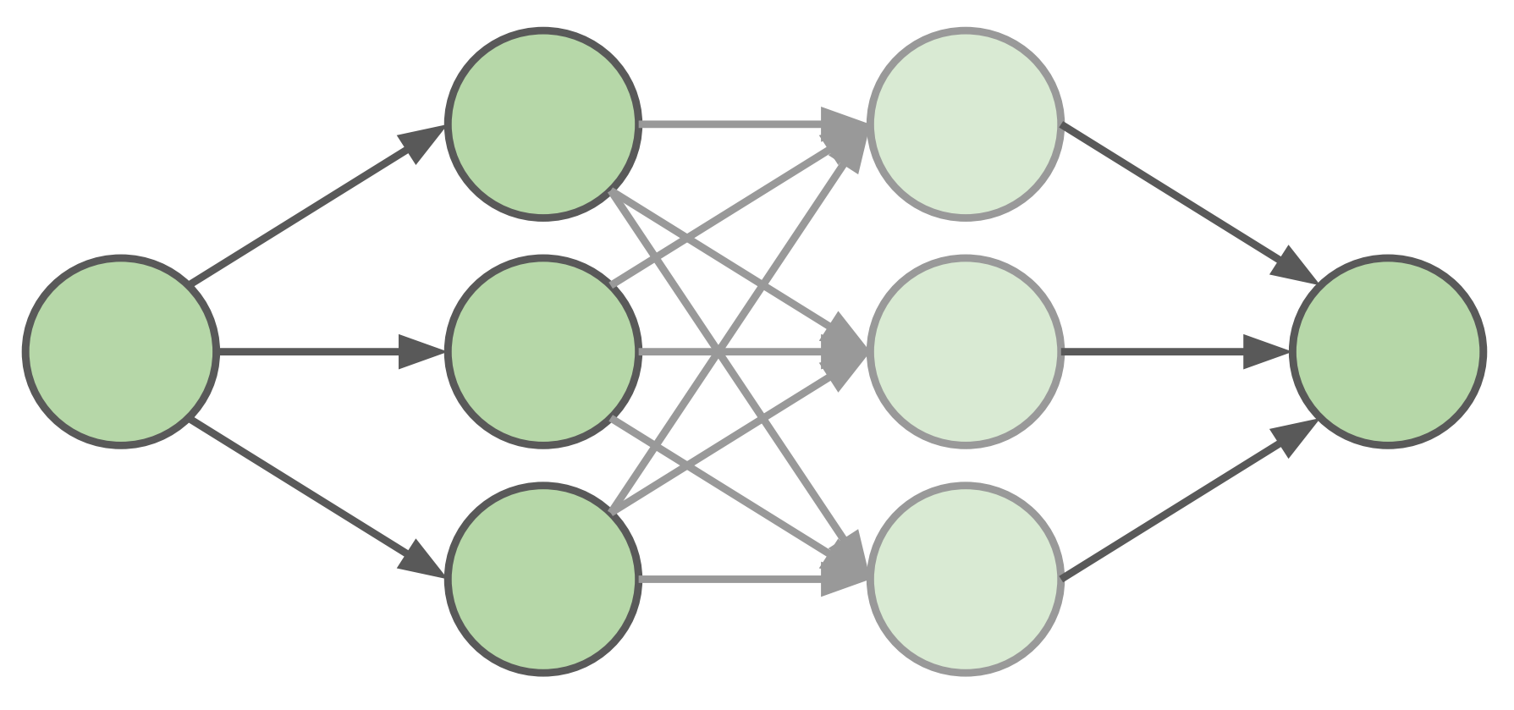
We would like a notion of "usefulness" for a parameter expansion which is meaningful even if the overall function remains constant. One possibility is the simple magnitude of the gradient, as suggested by Evci et al.[1]. We obtain much better behaviour from our second order estimate of converged performance, avoiding issues like the addition of duplicate neurons or bias based on average activation magnitudes differing across layers and neurons.

Parameter Insertion

We would like to add capacity without changing the overall function.



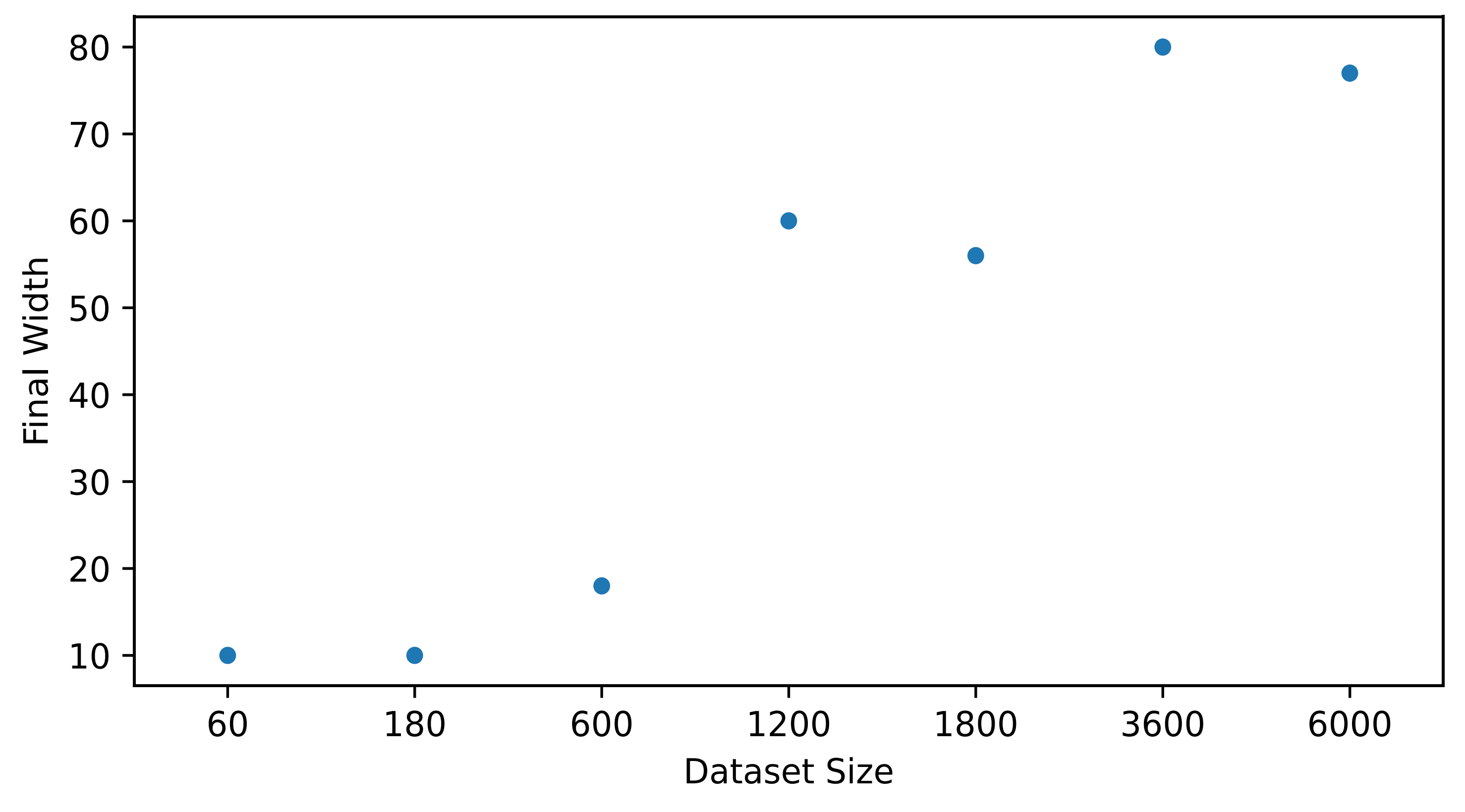
When we expand an existing layer we can avoid changing the overall function by initialising the output weights to zero, leaving the input weights as free parameters.



What about adding layers? If we choose a parameterised nonlinearity which can be set to the identity, then we can exploit invertibility of the resulting layer to avoid changing the function.

Choosing Width Appropriate to Dataset Size

Converged hidden size for a Single Layer Perceptron as a function of number of examples per class for the MNIST dataset. Note the log scale.



[1] Evci, U., van Merriënboer, B., Unterthiner, T., Vladymyrov, M., & Pedregosa, F. (2022). GradMax: Growing Neural Networks using Gradient Information. *ArXiv [Cs.LG]*. Retrieved from http://arxiv.org/abs/2201.05125: