having the weighting of the old, now disabled connection. While the implementation differ most architecture mutations follow a similar procedure, having a operation which randomly selects a node or connection, then removes, adds or changes the operation of that component. These system commonly employ a bank of operations from which a new operation can be selected [3], [32].

A similar approach which takes a slightly different direction is the use of *Network Morphisms* as described in LEMONADE. These are again transforms which are applied to the network, however, the goal of the operation is to deepen or wider the network while preserving the network function. A network morphism is a transform T, on a network N, which satisfies equation 8 for $x \in X$. This allows for the topology of the network to be expanded while maintaining the performance of the network. An example of this would be the addition of a layer to a network which is initialised to the identity function.

$$N^w(x) = (TN)^w(x) \tag{8}$$

explain

This type of network transform has an obvious advantage for the efficient use of weight inheritance also know as Lamarckian Inheritance. This process has similarities with some neuroevolutionary approaches, as the network weights are maintained through the process of mutation and the generation of offspring. This be considered a form of warm-starting as networks are effectively pre-trained with their previous structure.

Another approach to weight inheritance is Population Based Training (PBT) introduced by [23]. This is a GA which takes inspiration from bandit approaches to hyper-parameter optimisation. This was able to outperform human tuned networks on reinforcement learning and imagine classification tasks.

This is a system for hyper-parameter optimisation which utilizes a variant of the successive-halving[13] method used in HyperBand (see Sec 4.1) for early-stopping, A portion of the population abandon their search and copy the structure, hyper-parameters and weights of high performing networks. The hyper-parameters of this new replicated network are then mutated randomly to allow for more thorough examination of lucrative search spaces. This method of exploiting successful networks produces a schedule of hyper-parameters which change over the course of the training process rather than a static set. While this does offer the potential for a more dynamic training system, the size of the search space is increased dramatically as the number of hyper-parameter schedules is combinatorially larger than the original number of hyper-parameter sets.