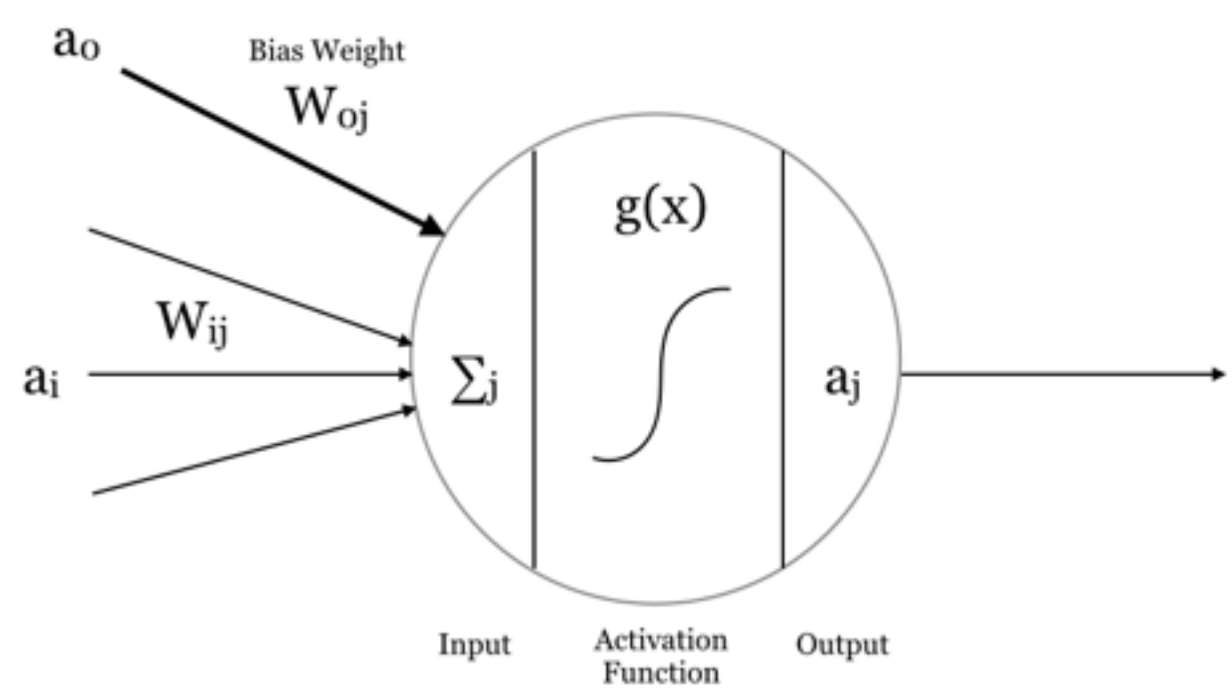
# Reducing the Complexity of Neural Networks

Samuel Jackson (slj11)

# NNs in a Nutshell

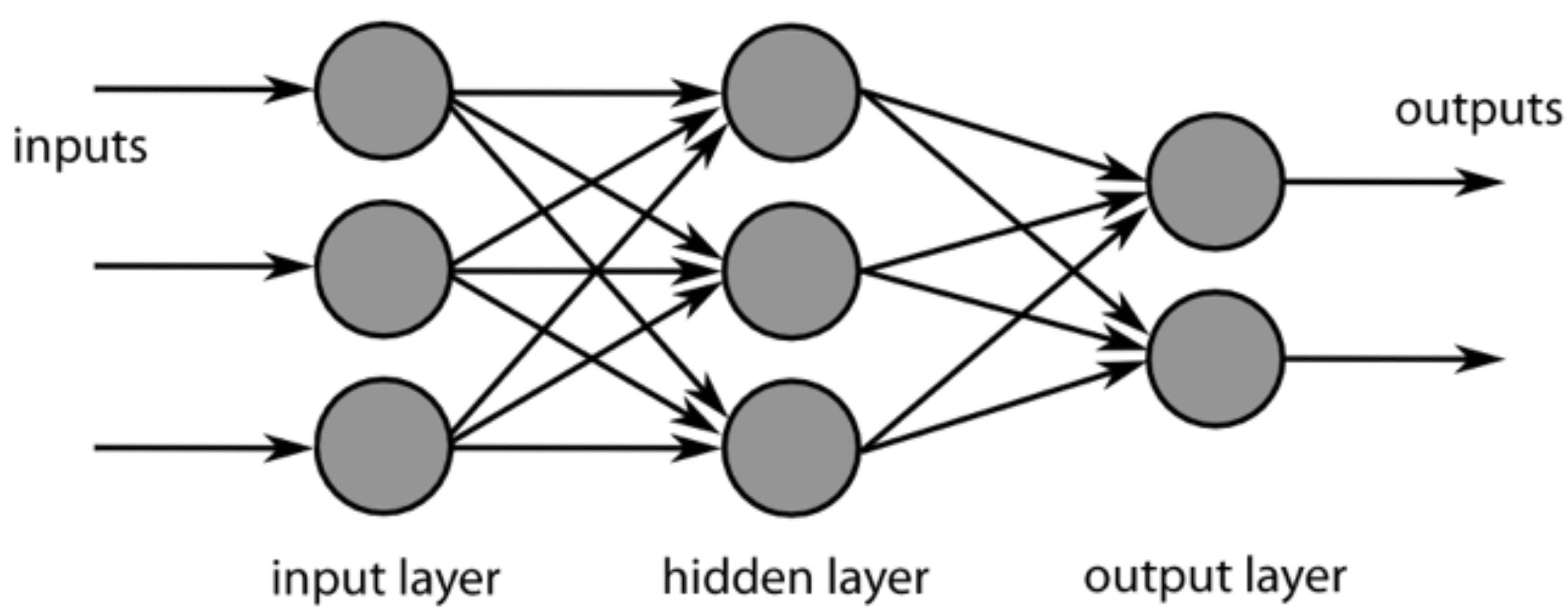
- Inspired by synapses in the brain.
- Idea has been around for over 70 years[1].
- Each perceptron takes a weighted vector of inputs.
- Activation function determines output.




1. McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

# NNs in a Nutshell

- Perceptrons stacked into layers.
- Typically will have multiple layers.
- Training:
  - Feed forward examples through the network.
  - Back propagate error between actual and expected.



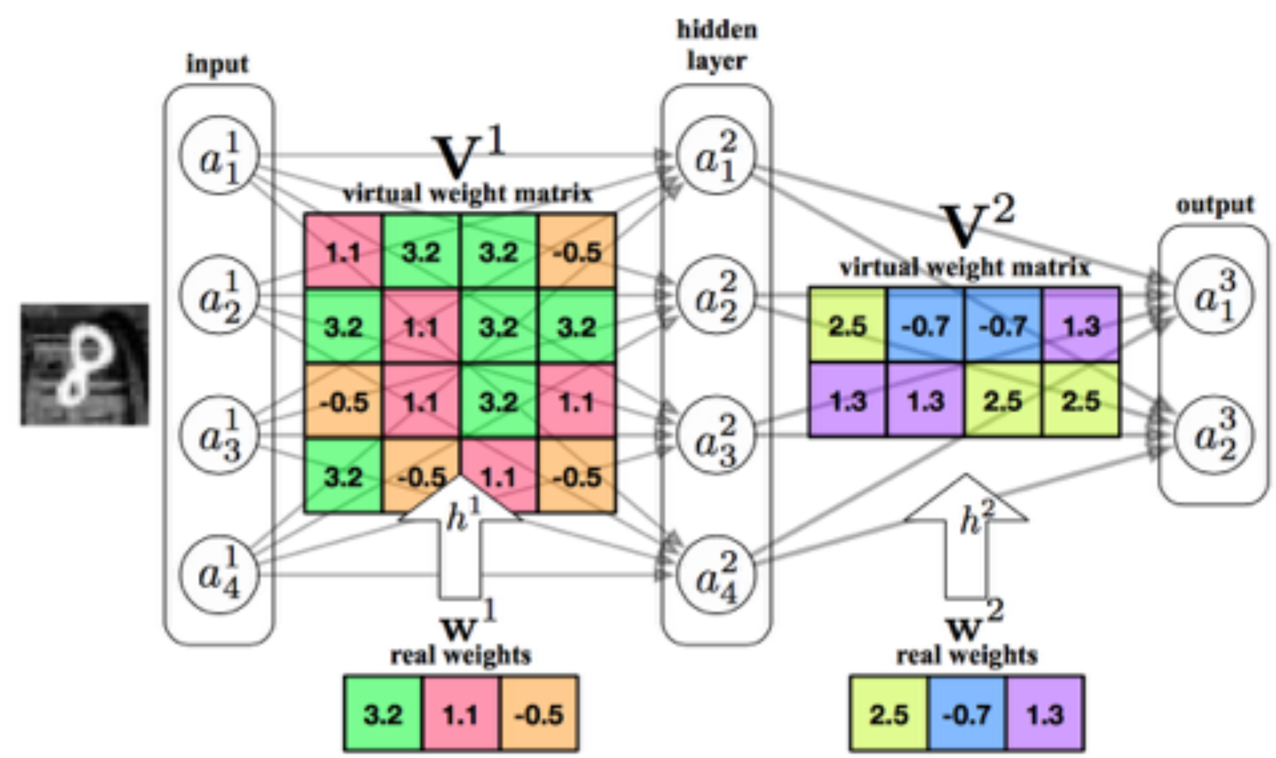
# Reducing Complexity

- Modern networks have 1000s of weights.
- Huge amount of time and memory used.
- Idea: Can we reduce or compress the number of weights?
- There can be a high amount of redundancy[1]
  - Sparsity: many weights are zero or nearly zero.

1. Denil, Misha, et al. "Predicting parameters in deep learning." *Advances in Neural Information Processing Systems*. 2013.

# HashedNets[1]

- Uses the “hash trick”: A hash function randomly maps virtual weights to an array of real weights.
- Size of real weight array determines the amount of compression.
- To save memory inputs are hashed rather than weights.



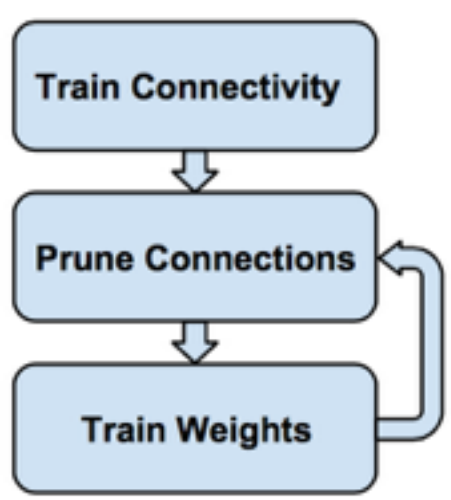

1. Chen, Wenlin, et al. "Compressing Neural Networks with the Hashing Trick." *arXiv preprint arXiv:1504.04788* (2015).

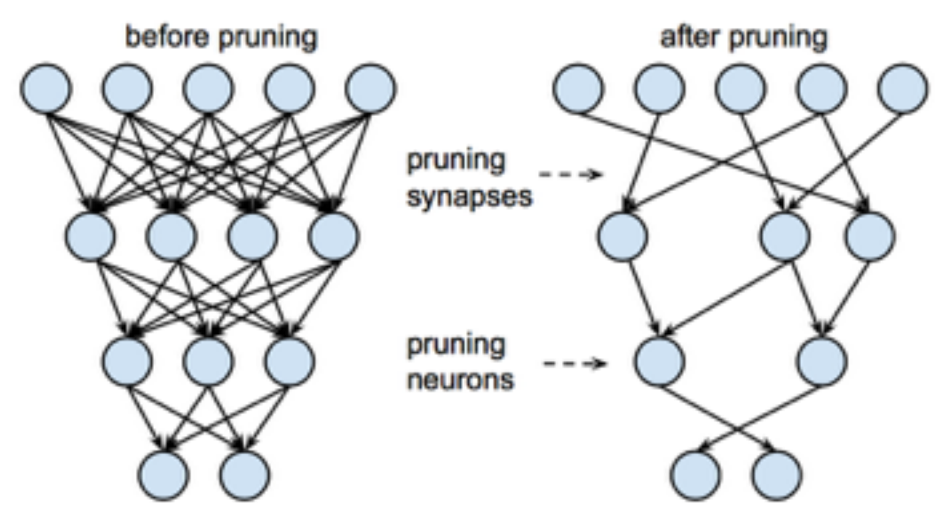
# HashedNets

- Pros
  - Can be trained using regular back propagation
  - Compression size is a fixed parameter
  - Is trained at a fixed size, rather than starting large and slimming down
  - Can easily be combined with other approaches
- Cons
  - Random memory access on GPUs is not efficient.
  - Compression size is a fixed parameter
  - No results with more general image set, or other architectures (e.g. Convolutional, Recurrent)

# Weight Pruning[1]

- Very simple idea:
  - Learn a full network
  - Prune connections with a threshold
  - Prune unconnected neurons
  - Retrain the pruned layers
- Similar to Dropout[2] but with without probabilities.




1. Han, Song, et al. "Learning both weights and connections for efficient neural networks." *arXiv preprint arXiv:1506.02626* (2015).
2. Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

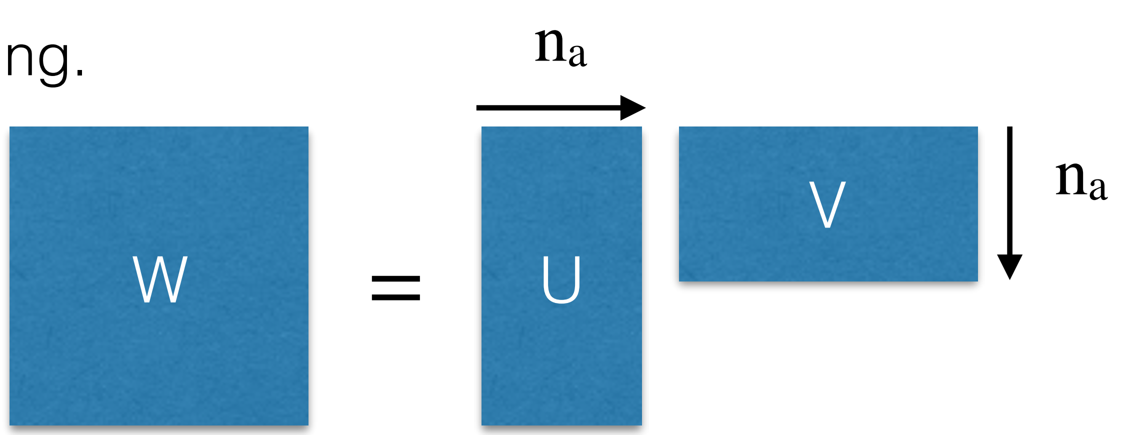
# Weight Pruning

- Pros
  - Quite a simple technique
  - Number of weights selected automatically
  - Authors used Caffe[1] and well known architectures so results are very reproducible.
  - "Better than random"
- Cons
  - Number of weights selected automatically.
  - Requires growing a full network architecture first.

1. Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014.

# Low-Rank Decomposition[1]

- Weight matrix is often very big ($n_v$ x $n_h$)
- Can be decomposed into two smaller matrices
  - $\mathbf{W} = \mathbf{UV}$
  - Where $\mathbf{U}$ is of size $n_v$ x $n_a$ and $\mathbf{V}$ is of size $n_a$ x $n_h$
  - $n_a$ Is parameter determining the space saving.




- Keep $\mathbf{U}$ fixed and learn only $\mathbf{V}$
  - $\mathbf{U}$ becomes a "dictionary of bases"
  - $\mathbf{V}$ becomes the parameters for linear combinations of $\mathbf{U}$ representing learned features.
- Many, many choices for $\mathbf{U}$ discussed by the authors.

1. Denil, Misha, et al. "Predicting parameters in deep learning." *Advances in Neural Information Processing Systems*. 2013.

# Low-Rank Decomposition

- Pros
  - Many choices for $\mathbf{U}$
  - Authors propose an architecture that can use multiple $\mathbf{U}$
  - Doesn’t require a full network to be learned.
  - Independent of the choice of activation, regularisation, layer architecture etc.
- Cons
  - Complicated to implement
  - Choice of $\mathbf{n_a}$ is not obvious.
  - Choice of is $\mathbf{U}$ often not obvious

# Thank You

Any Questions?