Neural Networks with Python and TensorFlow

Alexander J. Johnson

Manchester Metropolitan University

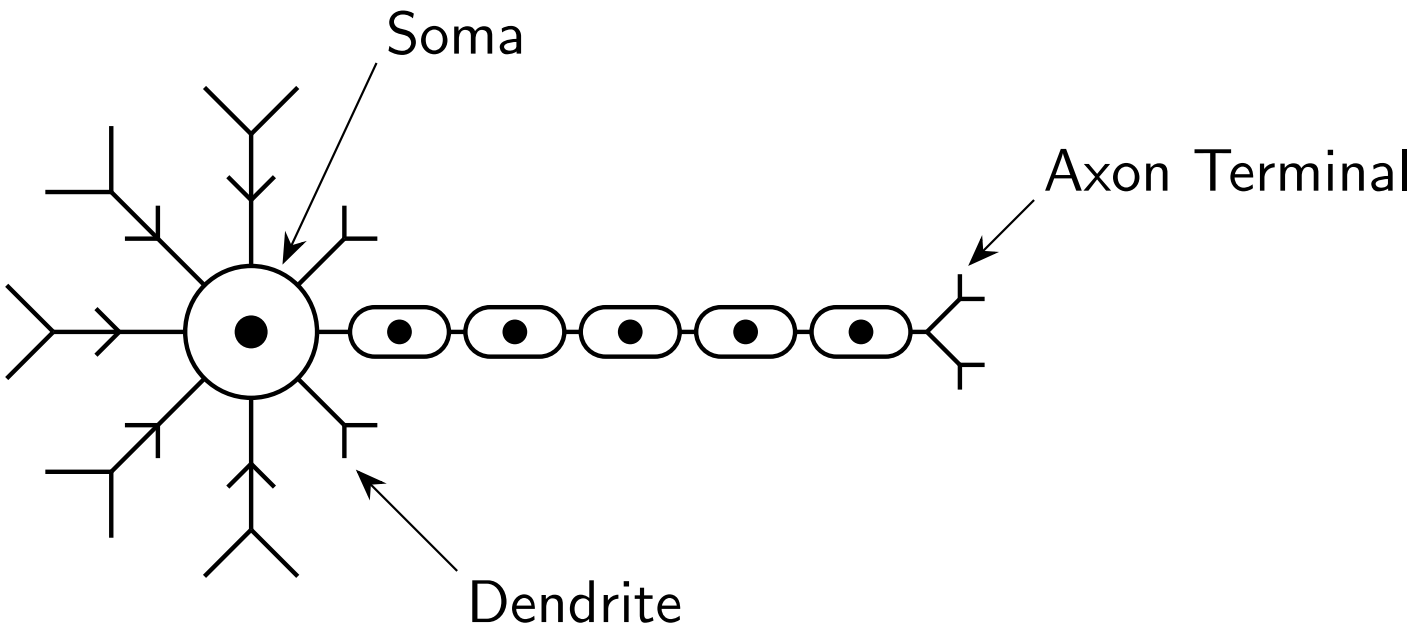
2020

Aims and Objectives

- Provide an introduction to a variety of neural network architectures.
- Implement the architectures in python using TensorFlow.
- Use neural networks to solve simple problems.

Basis

Biological Neuron:



Receives signals via dendrites and soma, applies threshold, outputs via axon terminal.

Perceptron

Rosenblatt, 1958:

$$A_i = \begin{cases} 1, & \sum_j w_{i,j} x_j > \theta \\ 0, & \text{otherwise} \end{cases}$$

Boolean output.
Weights adjusted according to activation and response correctness.

Modern Neural Network

Most common activation equation outlined by McClelland, Rumelhart, and Group, 1986:

$$y_i = \phi \left(b + \sum_j w_{i,j} x_j \right).$$

Weights adjusted via backpropagation / chain rule:

$$w_i' = w_i + \eta \frac{\partial f}{\partial w_i}.$$

Supported out-of-the-box by TensorFlow.

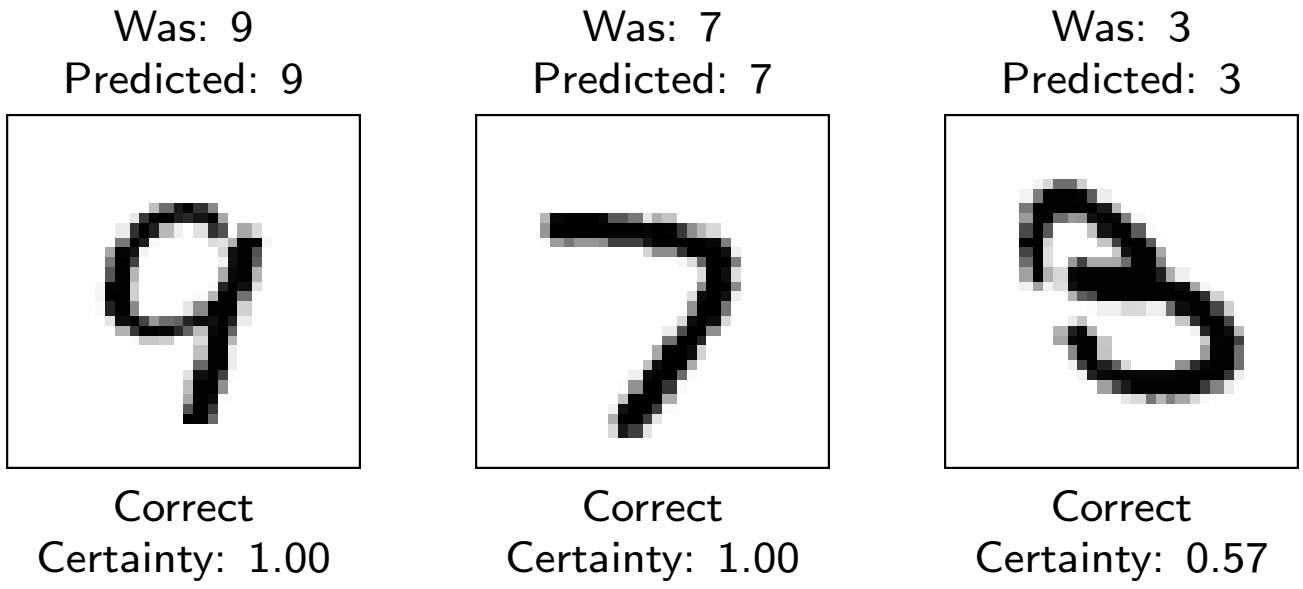
Convolutional

Original concept by Fukushima, 1980.
Features of convolutional layers:

- Neurons only connect to regions of previous layers.
- Share same weight scheme.
- Implicitly encodes position.
- Shift invariant, useful in image processing.

Supported out-of-the-box by TensorFlow.

Convolutional



Character recognition for the MNIST dataset using a convolutional neural network.

Recurrent

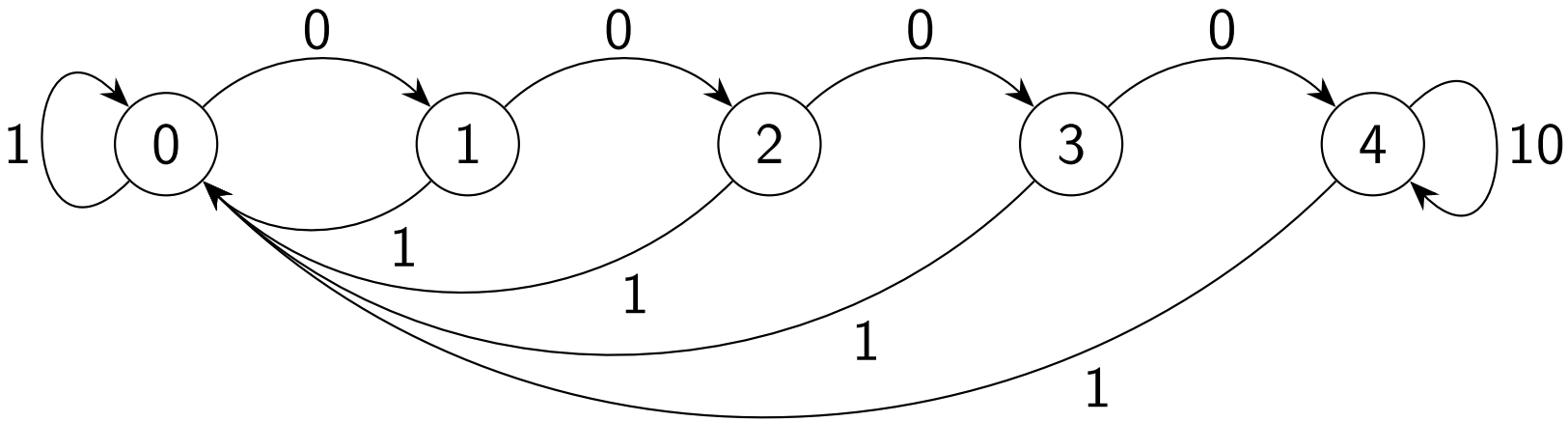
Features of recurrent layers:

- Has an internal state.
- Useful for temporal tasks.
- Suitable for both short and long time scales.

Supported out-of-the-box by TensorFlow.

Reinforcement Learning

Example: Chain problem from Strens, 2000.



Reinforcement Learning

Q-Learning (Watkins, 1989):

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Deep Q-Learning replaces table with network.
Improvements:

- Prioritised Experience Replay.
- Fixed Q-Targets.
- Double Deep Q-Networks.
- Dueling Deep Q-Network.

Implementing Dueling Deep Q-Networks requires some additional work, but is relatively simple.

Reinforcement Learning

Policy Gradient (Sutton et al., 2000):

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a)$$

Stochastic policy instead of deterministic.
Could not implement in TensorFlow during project due to limited informal resources about lower-level functions.
Can be combined with Deep Q-Learning to create Actor-Critic methods, which are currently the start-of-the-art method for reinforcement learning.

Conclusions

- Neural networks can be applied to a large variety of problems.
- TensorFlow provides an easy-to-use framework for defining and training models.
- Although well documented, TensorFlow's lower-level features and functions lack sufficient informal resources.

Source code:
`https://github.com/RanfordS/2020-Masters-Project`

Bibliography

Fukushima, Kunihiko (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36.4, pp. 193–202.

McClelland, James L, David E Rumelhart, PDP Research Group, et al. (1986). "Parallel distributed processing". In: *Explorations in the Microstructure of Cognition* 2, pp. 216–271.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386. DOI: 10.1037/h0042519.

Strens, Malcolm J. A. (2000). "A Bayesian Framework for Reinforcement Learning". In: *ICML*.

Sutton, Richard S et al. (2000). "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems*, pp. 1057–1063.

Watkins, Christopher John Cornish Hellaby (1989). "Learning from delayed rewards". In: