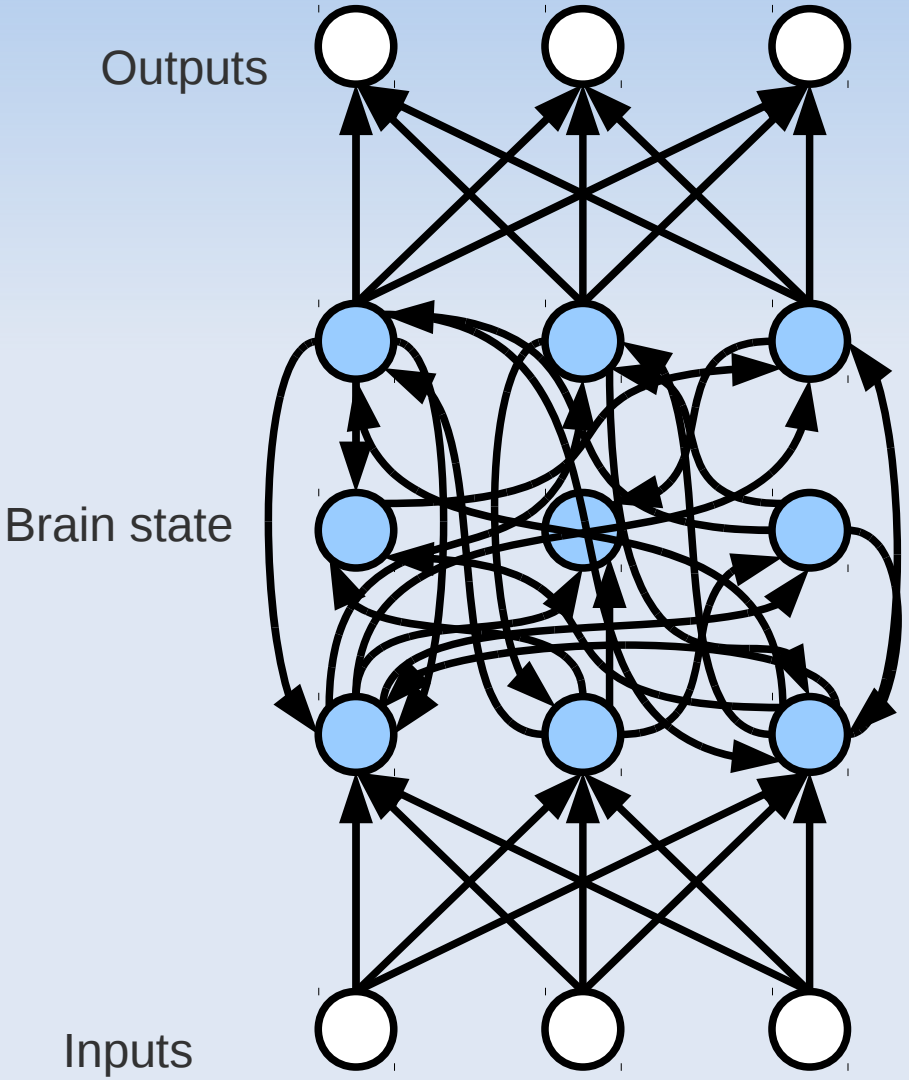
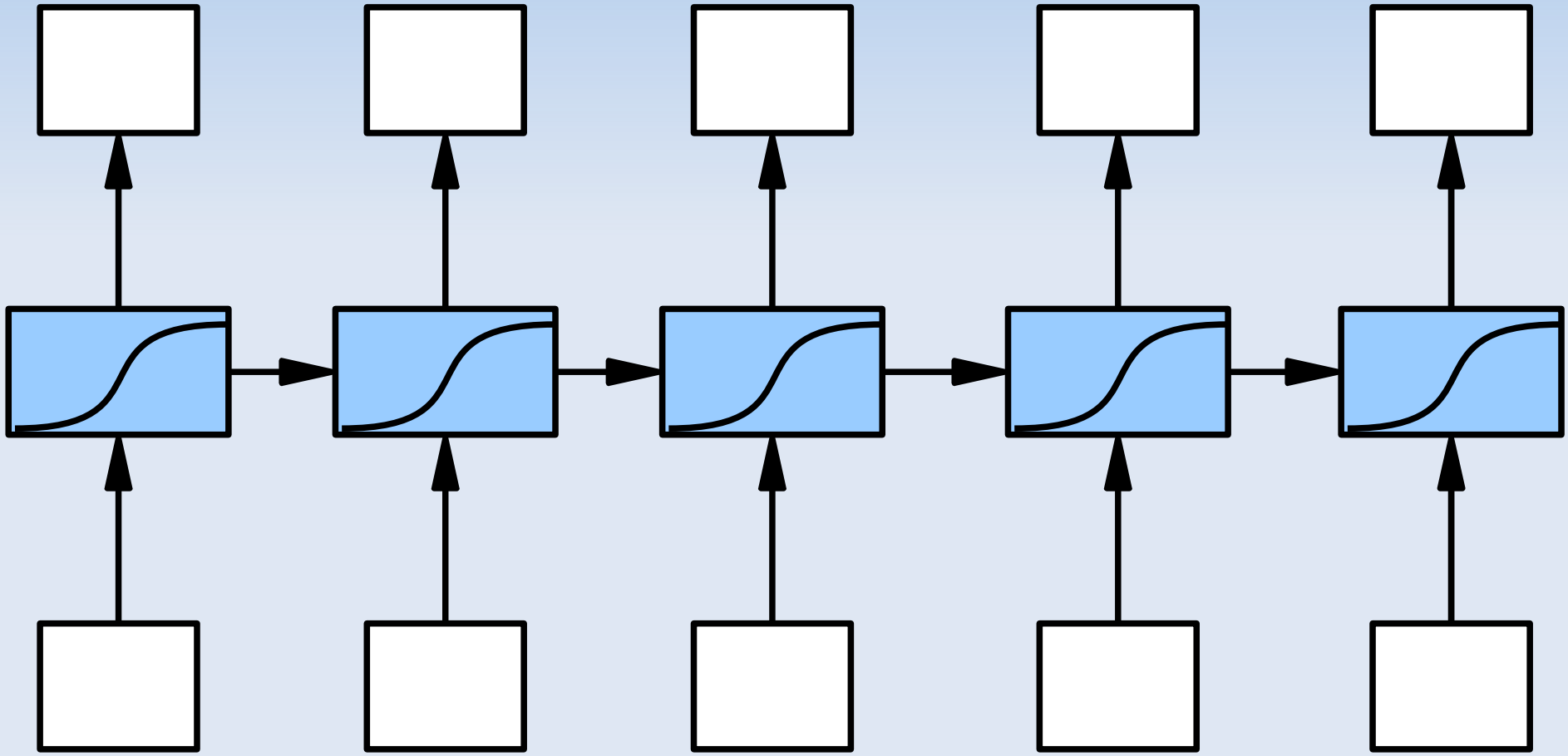
Training Recurrent Neural Networks to do cool stuff

Ilya Sutskever
James Martens
Geoff Hinton

Recurrent Neural Networks



Recurrent Neural Networks



Large hidden states

Rich dynamics

Recurrent Neural Networks

- They are much like small brains
- And we know what large brains can do
- We will see what small brains can do

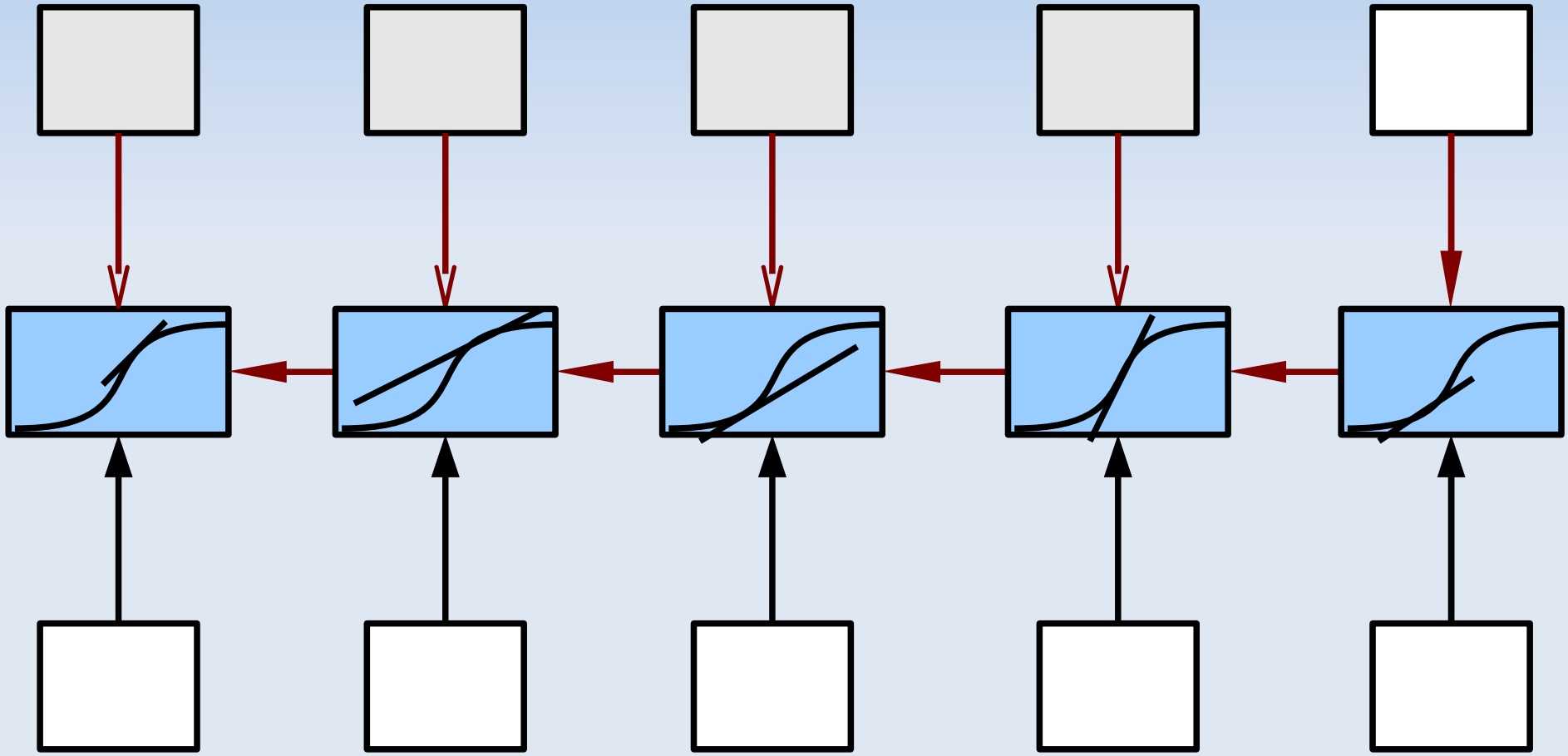
Training RNNs

- Big deal, can't we optimize the training error?
- Backprop through time easily computes the gradients of the RNN
- What's wrong with gradient descent?

Training RNNs

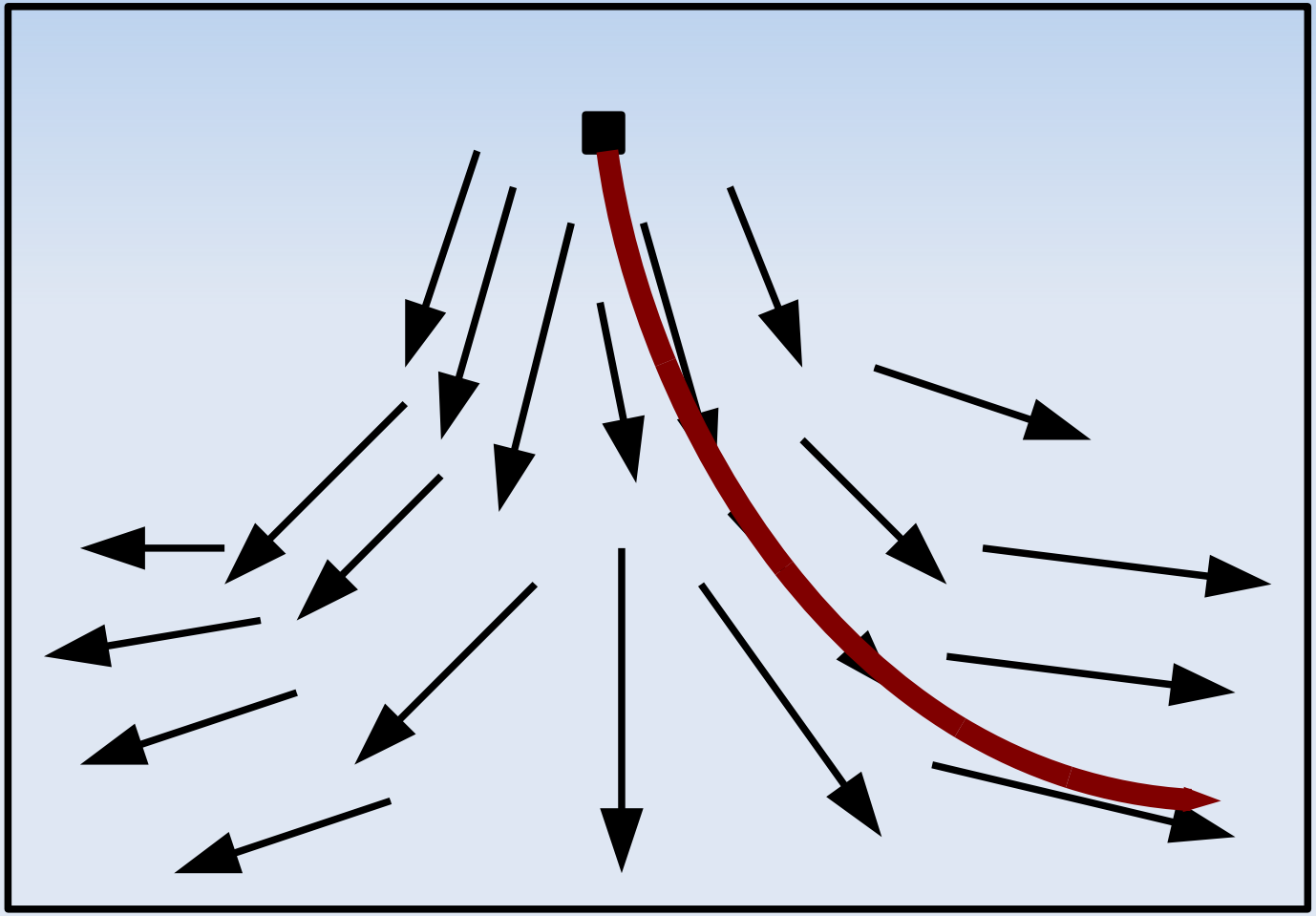
- Gradient descent simply doesn't work
- Just like deep learning is hard
  - Multiple iterated non-linearities
- The RNN is often extremely sensitive to small changes in its parameters
  - The exploding gradient problem
  - Ie, the butterfly effect

Backprop through time

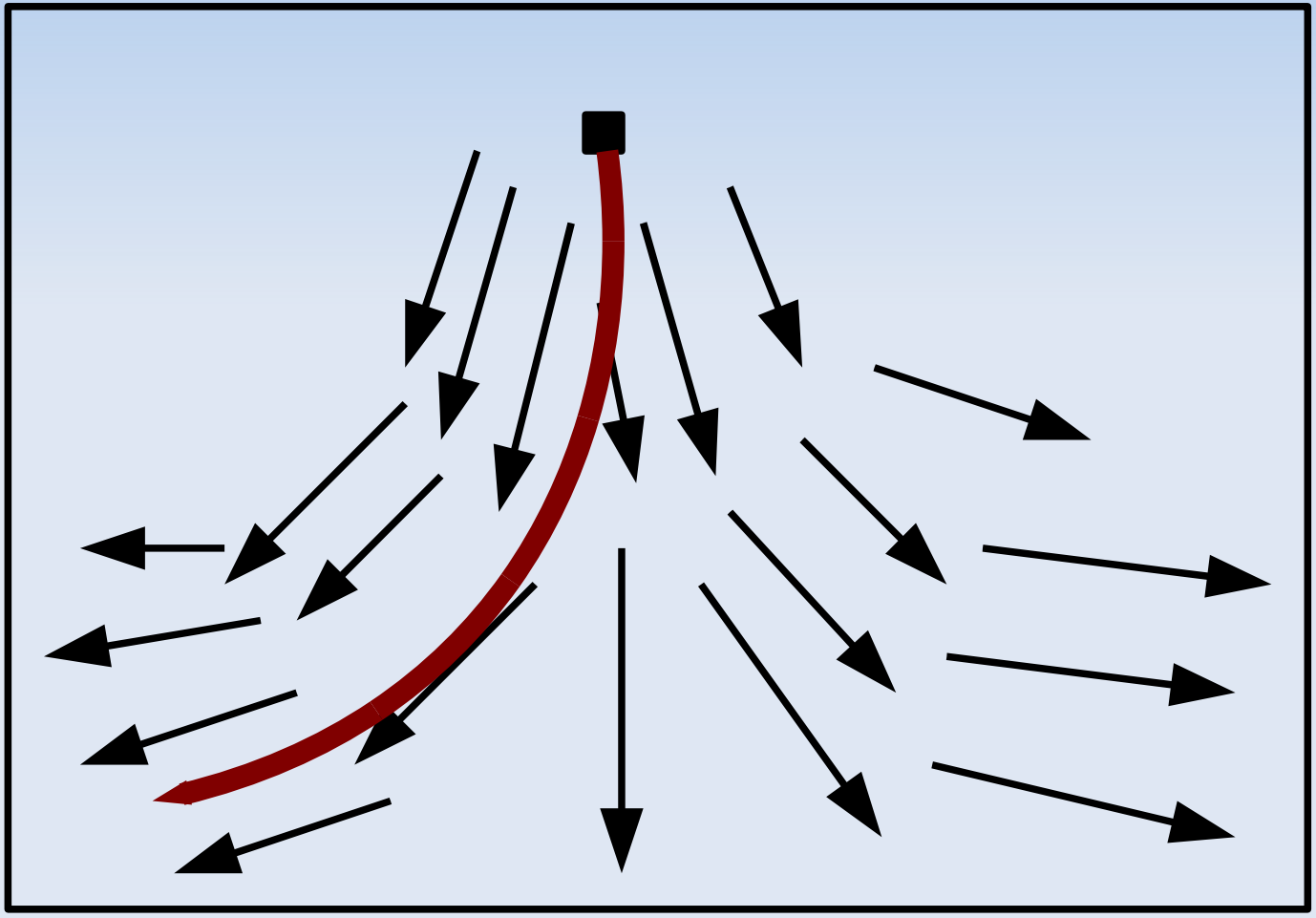


Iterated matrix multiplication is exponential: either shrink or blow up

Extreme sensitivity example



Extreme sensitivity example



Training RNNs

- So training RNNs is difficult
  - For gradient descent
- But what if we use the Hessian-free optimizer?
- Can it utilize the great expressive power that RNNs posses?

The real goal of the talk

- Is to calculate a sum

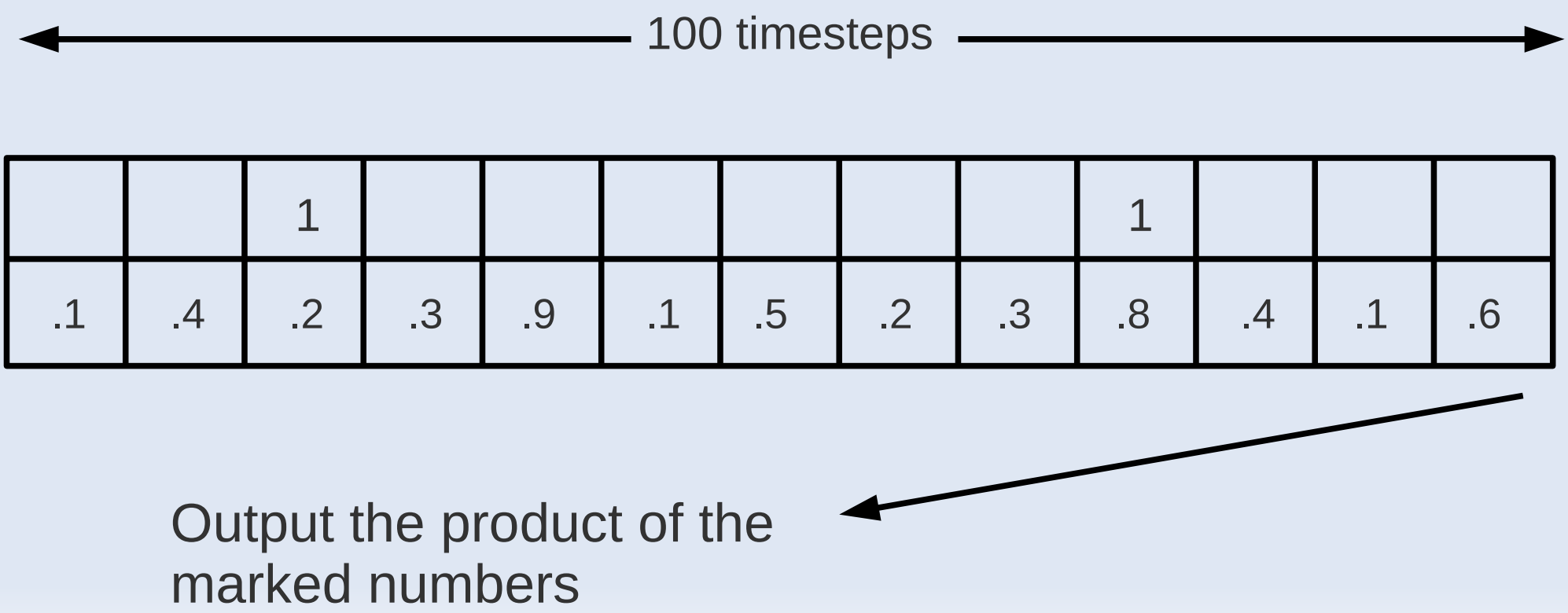
Rich, expressive, hard to use model
+
a very powerful optimizer

The experiments

- Tasks with pathologically long term dependencies
  - A task much harder than “straight memorization”
- Predict the next character in a stream of text using plenty of context
- Predict and Generate music

Extreme long term dependencies

- The multiplication problem [LSTM]



Hardness of multiplication

- The RNN needs to remember the marked numbers with high precision
- It also needs to completely ignore the unmarked numbers that bombard its hidden state
- Not a problem for our optimizer

Character-level language modelling

- Take Wikipedia
- Arrange its letters in a sequence
- Train an RNN to predict the next character from its past
  - Train on string fragments of 100 characters

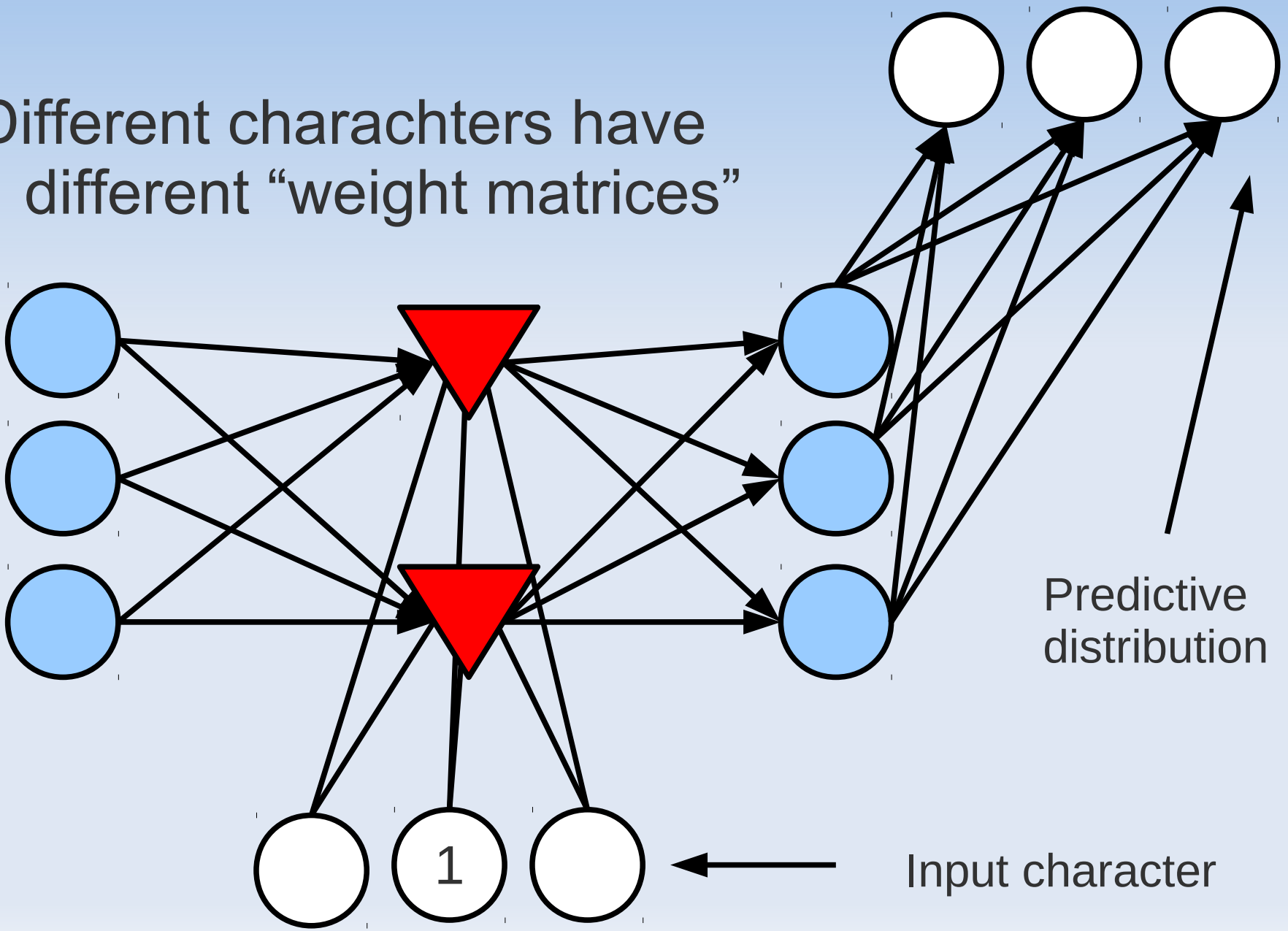
Specifics

- The RNN is trained to maximize the average log probability of the next character given its past

- We use a special multiplicative RNN

Specifics

- Different charachters have different “weight matrices”



Performance

4 gram	1.4 nats / char
PPM	1.1 nats / char
2000 unit RNN	1.02 nats / char

- Perplexity reduction: 244 to 164

Demo time

- Sample
  - The model is probabilistic but the training is completely deterministic
  - All the randomness is purely in the output units
- Debag
- Log prob demo, see file

Summary

- Little brains, or RNNs, are quite powerful
- Used to be untrainable

- The HF optimizer can train RNNs to solve very challenging tasks that involve significant long term dependencies

Thank you!

Music

- A midi file is a list of all the “notes”, with their beginning and end times
- Discretize time and represent the notes as binary vectors
- Learn to predict the notes played at the next timestep

How the data looks like