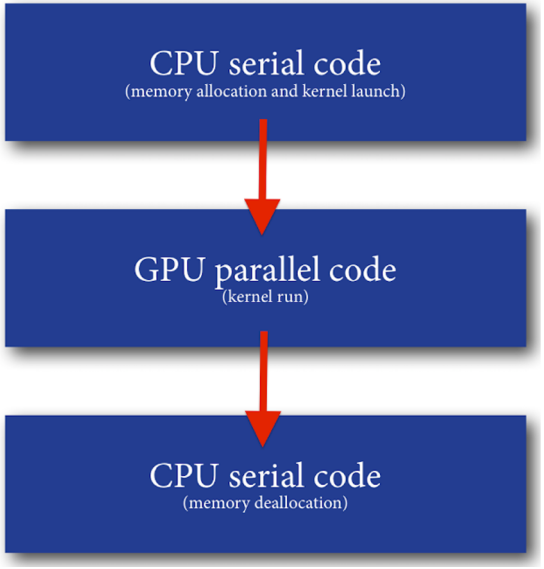
LSTM

By PG Cambie-Fabris,
Joshua Zingale

Outline

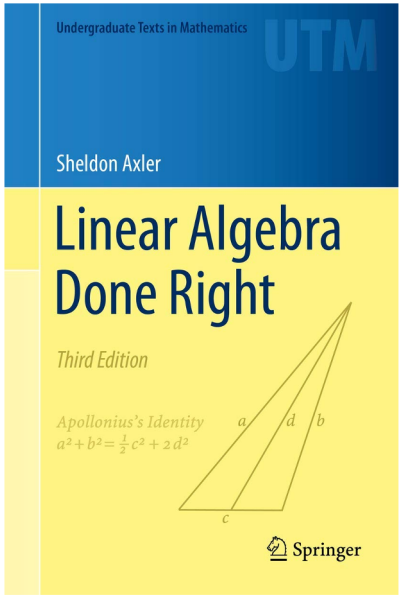
- CUDA
- MAGMA
- MagmaDNN
- Tensors
- Operations
  - Compute trees
- Layers
  - *LSTM*



Vector Addition

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

MAGMA

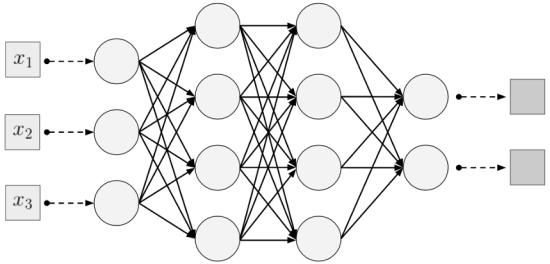

Undergraduate Texts in Mathematics
UTM
Sheldon Axler
Linear Algebra Done Right
Third Edition
Apollonius's Identity
$a^2+b^2=\frac{1}{2}c^2+2d^2$
a
d
b
c
Springer


NVIDIA
CUDA


NVIDIA

MagmaDNN

MagmaDNN > TensorFlow



Layer-based model building

C++

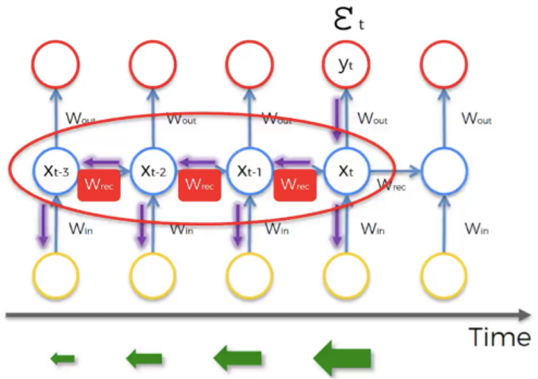
MAGMA

CUDA

Why LSTM?

0.9*0.9*0.9*0.9.... = 0 ; 1.1*1.1*1.1....= inf

The Vanishing Gradient Problem



$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta} \quad (3)$$

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right) \quad (4)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1})) \quad (5)$$

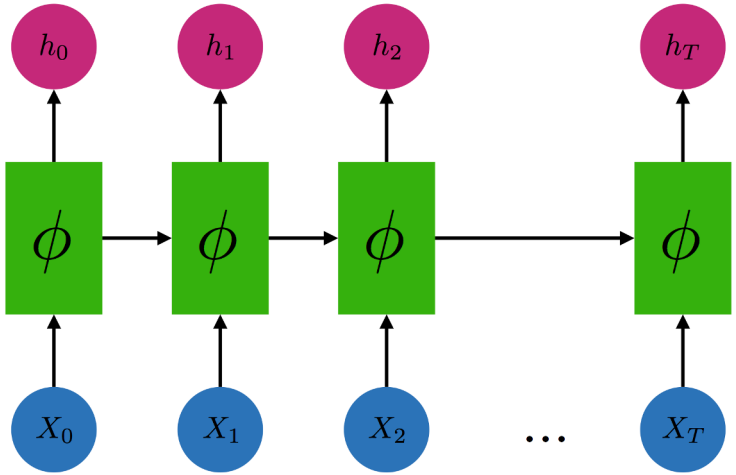
W_{rec} ~ small ⇨ Vanishing

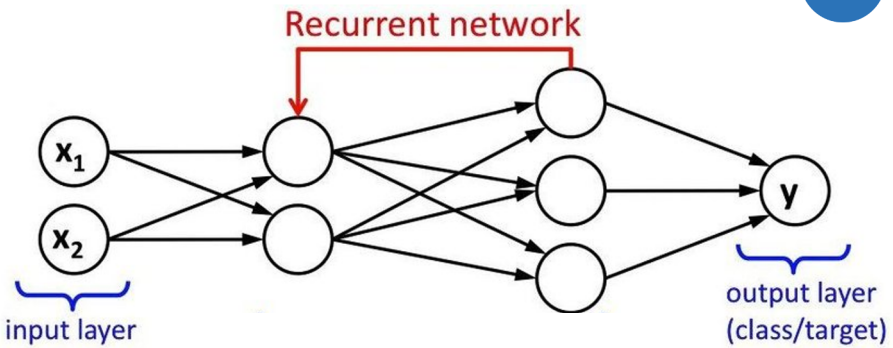
W_{rec} ~ large ⇨ Exploding

Formula Source: Razvan Pascanu et al. (2013)

What is LSTM?

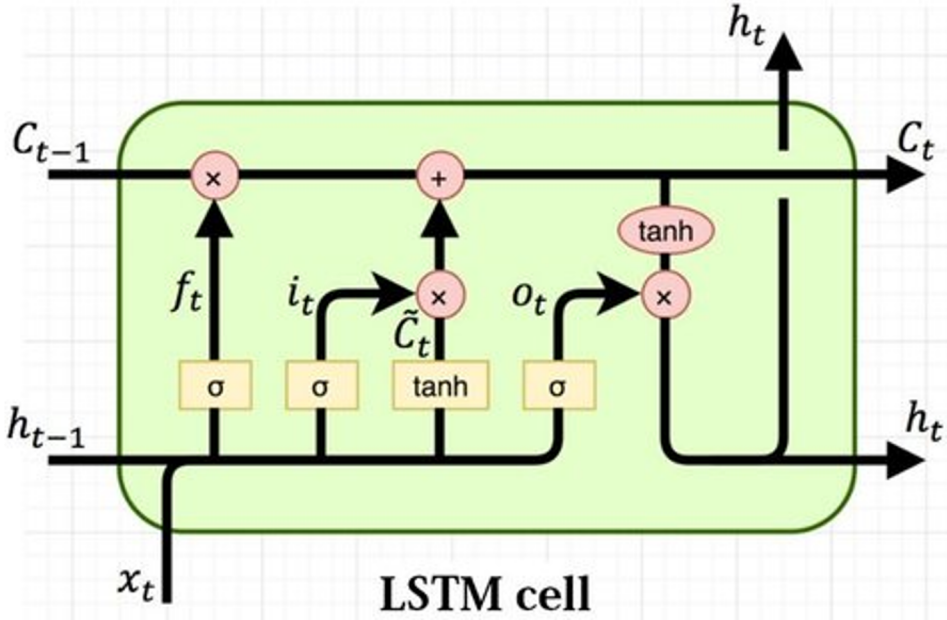
- Long Short-Term Memory
- Recurrent Neural Network





LSTM Uses


#906wx
STORM TEAM 3
EXTENDED FORECAST
UPMATTERS.COM
DAN PERKINS CONSTRUCTION
MON
TUE
WED
THU
FRI
SAT
SUN
20s & 30s PM WINTRY PRECIP
30°-30s COLDER SUPERIOR
30s-40° AM SNOW PM SUN
40°-40s RAIN/SNOW CHANCE
FEW SNOW SHOWERS
30°-30s FEW SNOW SHOWERS
30°-30s FEW SNOW SHOWERS
NNW 10+
NNW 5-15
S 5-10
WSW 10+
WNW 10+
NNE 5-15
NE 5-15
30s
30s
30s
40s
30s
30s
30s
10s
20s
20s
20s
20s
20s



LSTM cell

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$

$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$

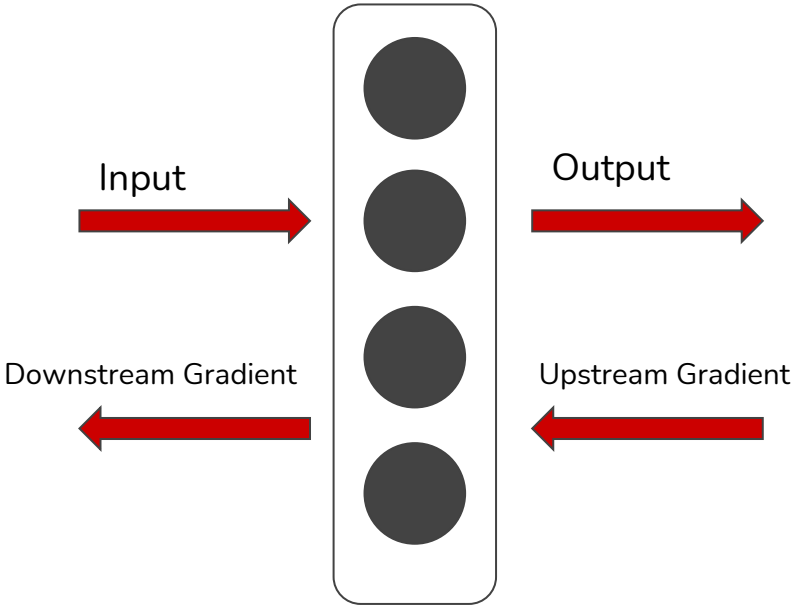
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$

$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$

$$h_t = \tanh(C_t) * o_t$$

LSTM Implementation Requirements

- Forward Pass
- Backward Pass



First Implementation

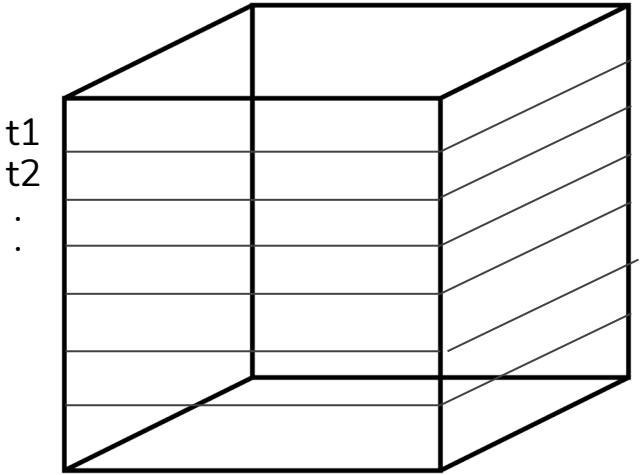
- Combine pre-existing operations with new ones
- CPU and GPU
- Forward and Backward method
    - Each individual operations’ eval and gradient

Needed Operations

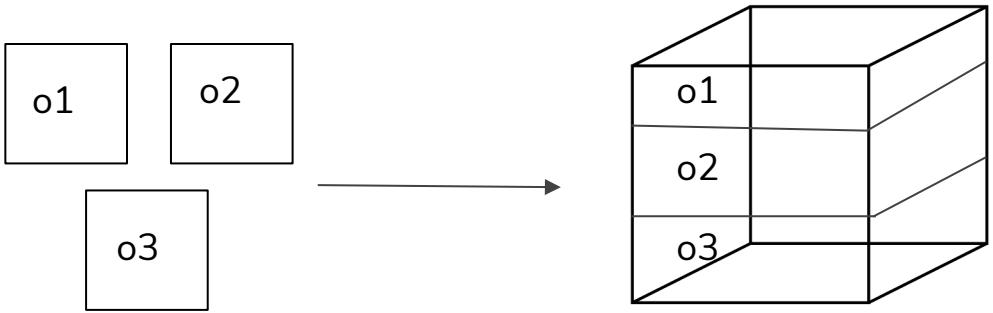
- Currently implemented
  - Sigmoid
  - Tanh
  - Matrix Multiplication
  - Matrix Addition
  - Element-wise Product
- Not Implemented
  - Slice
  - Concatenation

Slice and Concat

- Slice



- Concat



Slice and Concat Testing

- Both implemented on CPU and GPU
- Forward and Backward tested with examples on both
  - Slice/Concatenate passed in tensor and compare output
  - Pass in upstream gradient and compare downstream

```
Tensor size of {1, 2, 3, 4}
{
  {
    | 0.00008,   1.31538,   7.55605,   4.58650, |
    | 5.32767,   2.18959,   0.47045,   6.78865, |
    | 6.79296,   9.34693,   3.83502,   5.19416, |
  }
  {
    | 8.30965,   0.34572,   0.53462,   5.29700, |
    | 6.71149,   0.07698,   3.83416,   0.66842, |
    | 4.17486,   6.86773,   5.88977,   9.30437, |
  }
}

Below is the given tensor sliced about axis 1 and by index 2 on GPU
Tensor size of {1, 1, 2, 4}
{
  {
    | 6.79296,   9.34693,   3.83502,   5.19416, |
    | 4.17486,   6.86773,   5.88977,   9.30437, |
  }
}
```

Problems in Development cont.

user1@REU1902-HP-Z800-Workstation: ~/magma/magmadnn/src/compute/reducesum

```
145         result = new Tensor<T>(bprops[0]->get_shape(), {NONE, {}},
146                                bprops[0]->get_memory_type());
147         math::sum(bprops, result);
```

Add these two lines

user1@REU1902-HP-Z800-Workstation: ~/magma/magmadnn/src/compute/reducesum

```
29      for (unsigned int r = x_idx; r < n_rows; r += x_stride) {
30          for (unsigned int c = y_idx; c < n_cols; c += y_stride) {
31              out[r * n_cols + c] = (axis == 1) ? grad[r] : grad[c];
32          }
```

We had to swap these two.

(the order shown in the image is correct)

First Implementation Analysis

Successes

- Calculations
- Training on small data

Shortcomings

- Adding operation overhead takes too much time for time sequences >10
  - Perhaps exponential growth
- Compute tree eval issues
- Can not train on large data

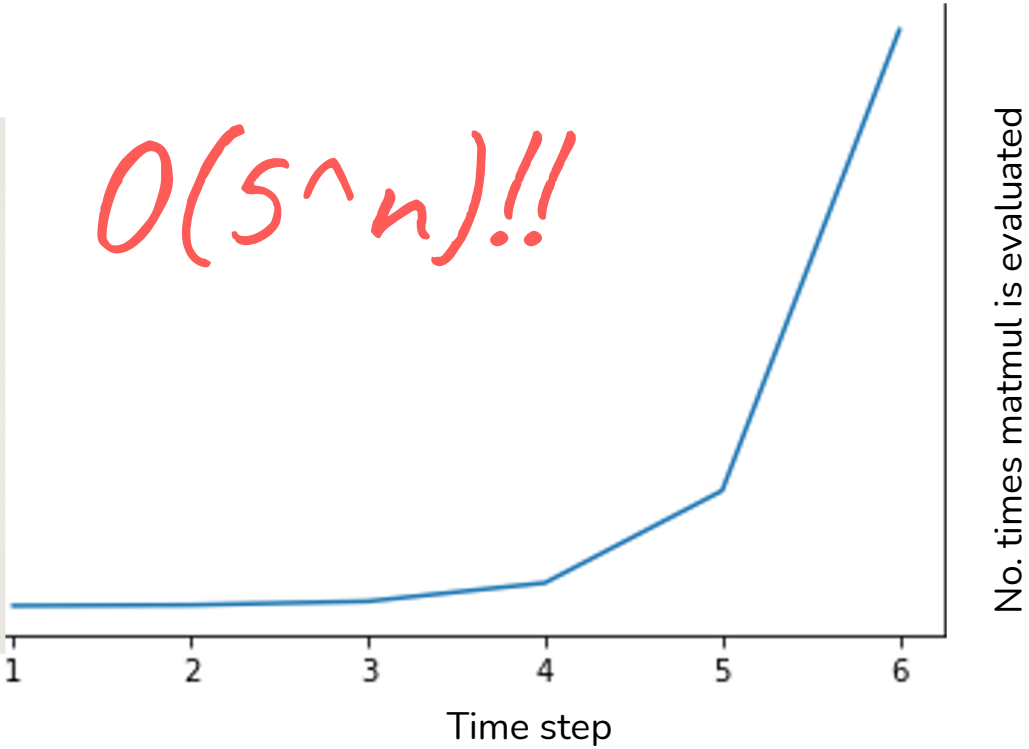
Operation Overhead

user1@REU1902-HP-Z800-Workstation: ~/magma/magmadnn/src/compute/reducesum

```
56 // TODO: With this uncommented, the simplelstm constructs exponentially slower at larger
57 // sequence lengths (~lengths >9)
58 // But with it commented, the GPU cannot work with any layer
59
60 #if defined(MAGMADNN_HAVE_CUDA)
61         // Use default stream for CUDA kernels
62         // this->custream_ = nullptr;
63         this->set_custream(nullptr);
64
65         this->set_cudnn_handle(magmadnn::internal::MAGMADNN_SETTINGS->cudnn_handle);
66
67         this->set_cublas_handle(magmadnn::internal::MAGMADNN_SETTINGS->cublas_handle);
68
69         this->set_async(false);
70 #endif
71     }
```

Compute Tree Reevaluation

O(5^n)!!



Second Implementation

- GPU only
- Use a single LSTM Operation
  - Cuda code from previous operations
  - New Cuda code for intermediate calculations
- Forward pass methodology
  - Kernels used
  - MAGMA sgemm and dgemm
  - Value caching
- Backward pass methodology
  - Kernels used
  - MAGMA sgemm and dgemm
  - Use of store values

Second Implementation Analysis

Resolved Issues

- Operation Overhead
- Eval issues

Ongoing Problems

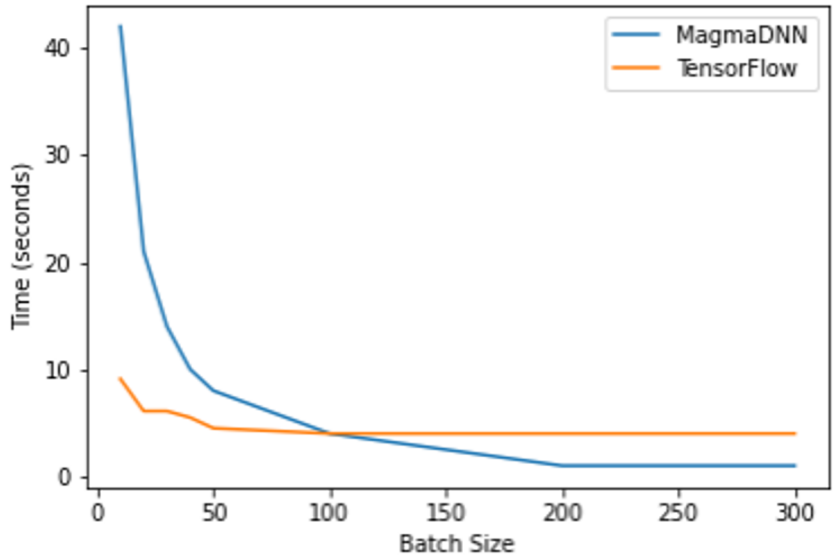
- Large data
- Runs out of memory

Testing

- Compared calculations against python script
  - Script verified against Dr. Wong test case
  - With and without return sequences
- Taught to predict zeros

```
NOTE: This is the result of a forward pass through the LSTMOp.
It should be 0.771982
Tensor size of {1, 1, 1, 1}
{
  {
    | 0.77198, |
  }
}

 NOTE: This is the gradient values calculated with respect to t
he inputs. It should be [[-0.05589539 -0.03405172], [-0.2005999
6, -0.12997548]]:
Tensor size of {1, 1, 2, 2}
{
  {
    |-0.05590,  -0.03405, |
    |-0.20060,  -0.12998, |
  }
}

 NOTE: This is the gradient values calculated with respect to W
f. It should be [[-0.01333808], [-0.08002848]]
Tensor size of {1, 1, 2, 1}
{
  {
    |-0.01334, |
    |-0.08003, |
  }
}

 NOTE: This is the gradient values calculated with respect to W
i. It should be [[-0.01152304], [-0.03248399]]
Tensor size of {1, 1, 2, 1}
{
  {
    |-0.01152, |
    |-0.03248, |
  }
}

 NOTE: This is the gradient values calculated with respect to W
o. It should be [[-0.12468496], [-0.71785738]]
Tensor size of {1, 1, 2, 1}
{
  {
    |-0.12468, |
    |-0.71786, |
  }
}
```

Performance



input -> lstm(5, return_sequences =true) -> lstm(1, false) -> output

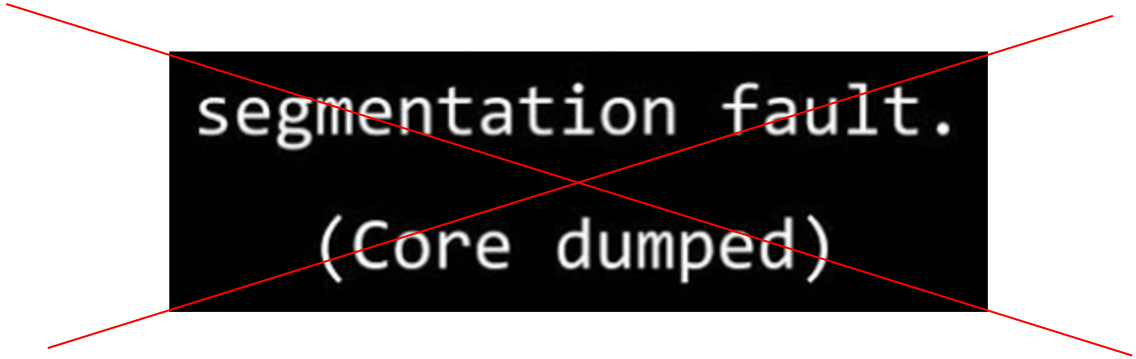
random initialization, all target values set to zero, 200 epochs, 300 input/output pairs

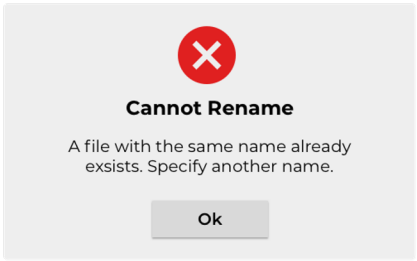
Future LSTM Work

- Understand the problem with the first implementation
  - i.e. why is does it get so slow? what makes it evaluate exponentially more operations?
- Fix memory issue with second implementation
- Add support for dynamically changing the input/output sequence length
  - This would likely require reworking the NeuralNetwork class

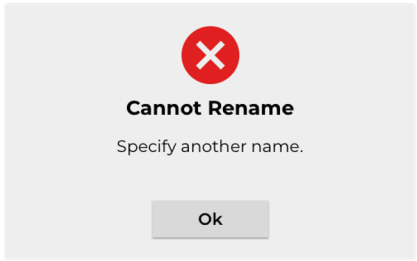
Future MagmaDNN Work

ADD ERROR MESSAGES





The message contains the problem, cause of error and solution.



The cause of the error is missing in this error message.

Future MagmaDNN Work cont.

- Abstract methods do not have clear descriptions of their responsibilities.