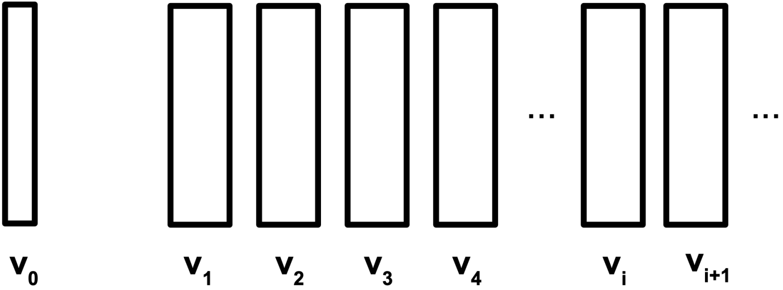
2nd Place @ ML Challenge #2
or:
How I ran out of Parameters

Daniel Lersch

November 20, 2019

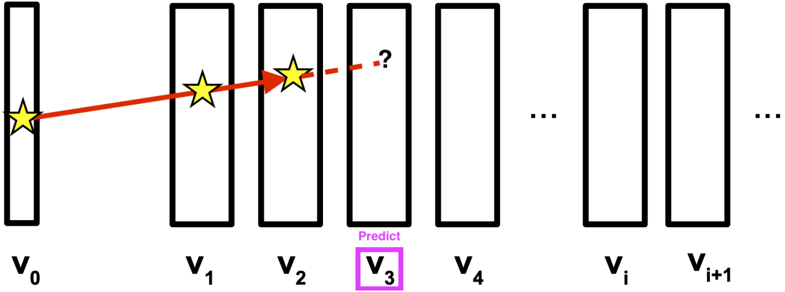
The Second ML Challenge

- GlueX **F**orward **D**rift **C**hamber (**FDC**)
  - 24 planes
  - Each plane: $\vec{v}_i = \begin{pmatrix} x \\ y \\ z \\ p_x \\ p_y \\ p_z \end{pmatrix}$
- **Goal:** Reconstruct particle in plane $i+1$, when all previous planes fired



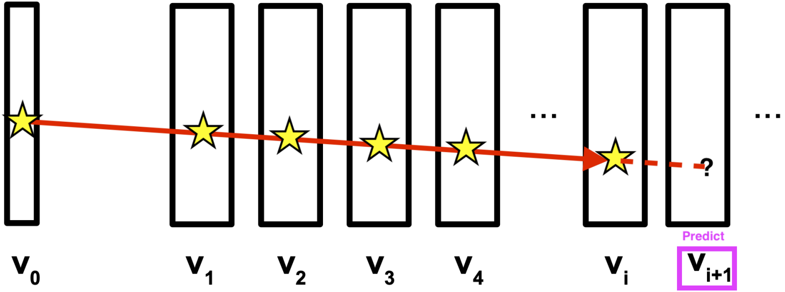
The Second ML Challenge

- GlueX **F**orward **D**rift **C**hamber (**FDC**)
  - 24 planes
  - Each plane: $\vec{v}_i = \begin{pmatrix} x \\ y \\ z \\ p_x \\ p_y \\ p_z \end{pmatrix}$
- **Goal:** Reconstruct particle in plane $i+1$, when all previous planes fired



The Second ML Challenge

- GlueX **F**orward **D**rift **C**hamber (**FDC**)
  - 24 planes
  - Each plane: $\vec{v}_i = \begin{pmatrix} x \\ y \\ z \\ p_x \\ p_y \\ p_z \end{pmatrix}$
- **Goal:** Reconstruct particle in plane $i+1$, when all previous planes fired



Preparing the Data for the Algorithm(s)

- **Strategy:** Introduce pattern into data, for ML algorithm to learn

Preparing the Data for the Algorithm(s)

- **Strategy:** Introduce pattern into data, for ML algorithm to learn
- Apply function f to each vector $\vec{v}_i$ in the sequence:

$$f(\vec{v}_i, t) = \begin{cases} \vec{v}_i, i < t, \\ \vec{\delta}_i, i == t, \\ \vec{\epsilon}_i, i > t \end{cases} \tag{1}$$

- with:

$$\vec{\delta}_i = \begin{pmatrix} \epsilon \\ \epsilon \\ z_t \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix}, \quad \vec{\epsilon}_i = \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix} \tag{2}$$

Preparing the Data for the Algorithm(s)

- **Strategy:** Introduce pattern into data, for ML algorithm to learn
- Apply function f to each vector $\vec{v}_i$ in the sequence:

$$f(\vec{v}_i, t) = \begin{cases} \vec{v}_i, i < t, \\ \vec{\delta}_i, i == t, \\ \vec{\epsilon}_i, i > t \end{cases} \tag{1}$$

- with:

$$\vec{\delta}_i = \begin{pmatrix} \epsilon \\ \epsilon \\ z_t \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix}, \quad \vec{\epsilon}_i = \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix} \tag{2}$$

- Using $\epsilon = 0$ leads to the sequence:

$$\vec{v}_0, \vec{v}_1, \vec{v}_2, ..., \vec{v}_{t-1}, \begin{pmatrix} 0 \\ 0 \\ z_t \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, ... \tag{3}$$

Preparing the Data for the Algorithm(s)

- **Strategy:** Introduce pattern into data, for ML algorithm to learn
- Apply function f to each vector $\vec{v}_i$ in the sequence:

$$f(\vec{v}_i, t) = \begin{cases} \vec{v}_i, i < t, \\ \vec{\delta}_i, i == t, \\ \vec{\epsilon}_i, i > t \end{cases} \tag{1}$$

- with:

$$\vec{\delta}_i = \begin{pmatrix} \epsilon \\ \epsilon \\ z_t \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix}, \quad \vec{\epsilon}_i = \begin{pmatrix} \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \\ \epsilon \end{pmatrix} \tag{2}$$

- Using $\epsilon = 0$ leads to the sequence:

$$\vec{v}_0, \vec{v}_1, \vec{v}_2, ..., \vec{v}_{t-1}, \begin{pmatrix} 0 \\ 0 \\ z_t \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, ... \tag{3}$$

- This is dangerous! $\Rightarrow$ Vanishing gradient!

Model Selection

- After a few test runs, decided to run with the following models:

$$M_1(n, u_0, u_1, .., u_{n-1}) = L_{lstm}(u_0) + \sum_{i=1}^{n-1} L_{dense}(u_i) + L_{dense}(6) \tag{4}$$

$$M_2(n, u_0, u_1, .., u_{n-1}) = L_{dense}(u_0) + \sum_{i=1}^{n-1} L_{dense}(u_i) + L_{dense}(6) \tag{5}$$

- with:

Variable	Meaning
L_{lstm}	LSTM recursive network layer
L_{dense}	Dense network layer
n	Number of hidden layers
u_i	Number of neurons in layer i

- NOTE: Input layer is not shown here
- Activation functions:
  - Output layer: Linear $\Rightarrow$ Want to regress the data
  - Every other layer: ReLU $\Rightarrow$ Have a lot of "0" in the data

Model Selection

- After a few test runs, decided to run with the following models:

$$M_1(n, u_0, u_1, .., u_{n-1}) = \mathbf{L_{lstm}}(u_0) + \sum_{i=1}^{n-1} L_{dense}(u_i) + L_{dense}(6) \tag{4}$$

$$M_2(n, u_0, u_1, .., u_{n-1}) = L_{dense}(u_0) + \sum_{i=1}^{n-1} L_{dense}(u_i) + L_{dense}(6) \tag{5}$$

- with:

Variable	Meaning
L_{lstm}	**LSTM recursive network layer**
L_{dense}	Dense network layer
n	Number of hidden layers
u_i	Number of neurons in layer i

- NOTE: Input layer is not shown here
- Activation functions:
  - Output layer: Linear $\Rightarrow$ Want to regress the data
  - Every other layer: ReLU $\Rightarrow$ Have a lot of "0" in the data

Model Selection

- After a few test runs, decided to run with the following models:

$$M_1(n, u_0, u_1, .., u_{n-1}) = L_{lstm}(u_0) + \sum_{i=1}^{n-1} \mathbf{L}_{\mathbf{dense}}(u_i) + \mathbf{L}_{\mathbf{dense}}(6) \quad (4)$$

$$M_2(n, u_0, u_1, .., u_{n-1}) = \mathbf{L}_{\mathbf{dense}}(u_0) + \sum_{i=1}^{n-1} \mathbf{L}_{\mathbf{dense}}(u_i) + \mathbf{L}_{\mathbf{dense}}(6) \quad (5)$$

- with:

Variable	Meaning
L_{lstm}	LSTM recursive network layer
L_{dense}	**Dense network layer**
n	Number of hidden layers
u_i	Number of neurons in layer i

- NOTE: Input layer is not shown here
- Activation functions:
  - Output layer: Linear $\Rightarrow$ Want to regress the data
  - Every other layer: ReLU $\Rightarrow$ Have a lot of "0" in the data

Model Selection

- After a few test runs, decided to run with the following models:

$$M_1(\mathbf{n}, u_0, u_1, .., u_{n-1}) = L_{lstm}(u_0) + \sum_{i=1}^{\mathbf{n}-1} L_{dense}(u_i) + L_{dense}(6) \tag{4}$$

$$M_2(\mathbf{n}, u_0, u_1, .., u_{n-1}) = L_{dense}(u_0) + \sum_{i=1}^{\mathbf{n}-1} L_{dense}(u_i) + L_{dense}(6) \tag{5}$$

- with:

Variable	Meaning
L_{lstm}	LSTM recursive network layer
L_{dense}	Dense network layer
n	**Number of hidden layers**
u_i	Number of neurons in layer i

- NOTE: Input layer is not shown here
- Activation functions:
  - Output layer: Linear $\Rightarrow$ Want to regress the data
  - Every other layer: ReLU $\Rightarrow$ Have a lot of "0" in the data

Model Selection

- After a few test runs, decided to run with the following models:

$$M_1(n, \mathbf{u_0}, \mathbf{u_1}, .., \mathbf{u_{n-1}}) = L_{lstm}(\mathbf{u_0}) + \sum_{i=1}^{n-1} L_{dense}(\mathbf{u_i}) + L_{dense}(6) \quad (4)$$

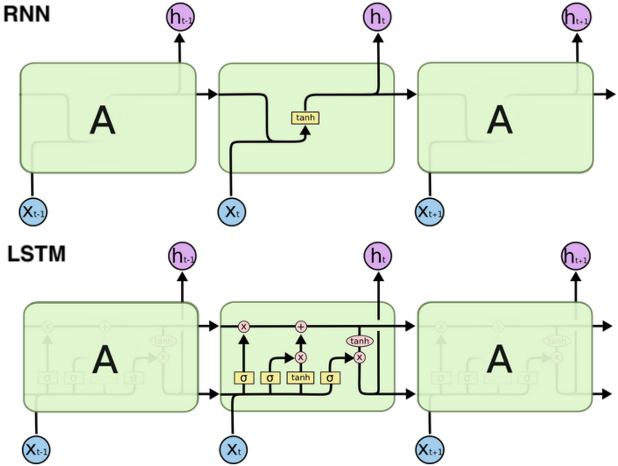
$$M_2(n, \mathbf{u_0}, \mathbf{u_1}, .., \mathbf{u_{n-1}}) = L_{dense}(\mathbf{u_0}) + \sum_{i=1}^{n-1} L_{dense}(\mathbf{u_i}) + L_{dense}(6) \quad (5)$$

- with:

Variable	Meaning
L_{lstm}	LSTM recursive network layer
L_{dense}	Dense network layer
n	Number of hidden layers
u_i	**Number of neurons in layer i**

- NOTE: Input layer is not shown here
- Activation functions:
  - Output layer: Linear $\Rightarrow$ Want to regress the data
  - Every other layer: ReLU $\Rightarrow$ Have a lot of "0" in the data

Short Reminder: RNN vs. LSTM

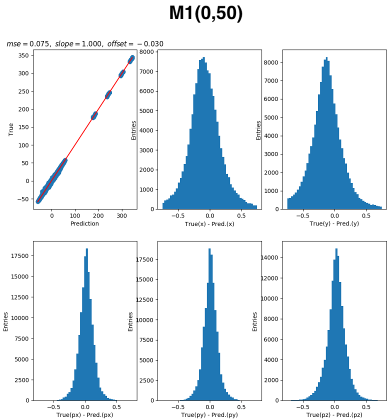


Also tested RNN, but LSTM showed better performance

Pictures taken from here (good explanation)

Interpretation and Training Strategy

- **Model 1:** Recurrent + regressor ⇒ Learn series (encoded in pattern) and fit data
- **Model 2:** Simple regressor ⇒ Simply fit the data, including the "0" pattern
- **Training strategy:**
  i) Train (and evaluate) several models on subset of training data ⇒ Save time
  ii) Re-train (and re-evaluate) "best" model on full training (validation) data

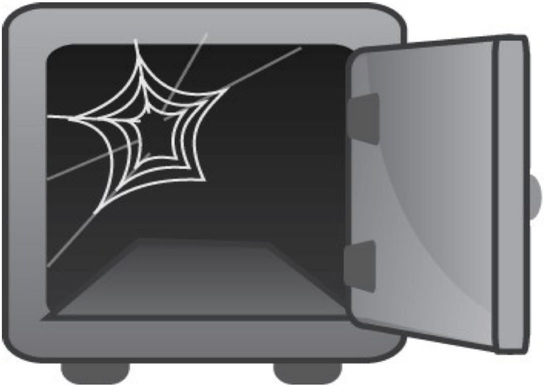


The Night before the Submission Deadline...

Everything was setup to train a deep neural net,

The Night before the Submission Deadline...

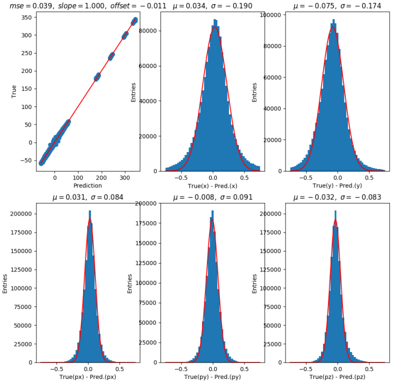
but...



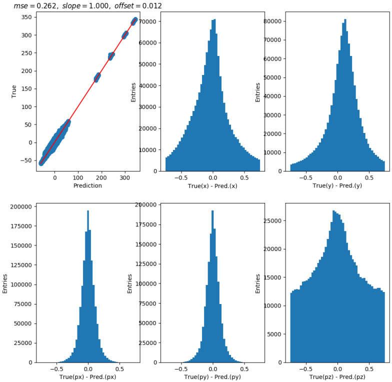
Final Configuration(s)

- $M_1 = L_{lstm}(50) + 2 \cdot L_{dense}(50) + L_{dense}(6)$ with $16.8\,\mathrm{k}$ parameters **(left)**
- $M_2 = 3 \cdot L_{dense}(50) + L_{dense}(6)$ with $10.4\,\mathrm{k}$ parameters **(right)**

M1



M2



Ideas and Tests that did not work out...

Picture taken from: http://screenrant.com/things-you-did-not-know-about-wile-e-coyote/

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$$\vec{v}_0, \vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{t-1}, \begin{pmatrix} n_{x,t} \\ n_{y,t} \\ z_t \\ n_{px,t} \\ n_{py,t} \\ n_{pz,t} \end{pmatrix}, \begin{pmatrix} n_{x,t+1} \\ n_{y,t+1} \\ n_{z,t+1} \\ n_{px,t+1} \\ n_{py,t+1} \\ n_{pz,t+1} \end{pmatrix}, \ldots$$

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$$\vec{v}_0, \vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{t-1}, \left(\begin{array}{c} n_{x,t} \\ n_{y,t} \\ z_t \\ n_{px,t} \\ n_{py,t} \\ n_{pz,t} \end{array} \right), \left(\begin{array}{c} n_{x,t+1} \\ n_{y,t+1} \\ n_{z,t+1} \\ n_{px,t+1} \\ n_{py,t+1} \\ n_{pz,t+1} \end{array} \right), \ldots$$

$\Rightarrow$ Worsened performance

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$\Rightarrow$ Worsened performance

- Changed order in model: lstm - dense - dense - .. $\rightarrow$ dense - lstm - dense - ...

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$\Rightarrow$ Worsened performance

- Changed order in model: lstm - dense - dense - .. $\rightarrow$ dense - lstm - dense - ...

$\Rightarrow$ Worsened performance $\Rightarrow$ Algorithm seemed to "forget"

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$\Rightarrow$ Worsened performance

- Changed order in model: lstm - dense - dense - .. $\rightarrow$ dense - lstm - dense - ...

$\Rightarrow$ Worsened performance $\Rightarrow$ Algorithm seemed to "forget"

- Introduced regularization

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$\Rightarrow$ Worsened performance

- Changed order in model: lstm - dense - dense - .. $\rightarrow$ dense - lstm - dense - ...

$\Rightarrow$ Worsened performance $\Rightarrow$ Algorithm seemed to "forget"

- Introduced regularization

$\Rightarrow$ No significant effect on performance

Ideas and Tests that did not work out...

- Introduced random noise $n_{k,i} = \mathcal{N}(0.0, \sigma_{k,i}) \neq 0$ to training data:

$\Rightarrow$ Worsened performance

- Changed order in model: lstm - dense - dense - .. $\rightarrow$ dense - lstm - dense - ...

$\Rightarrow$ Worsened performance $\Rightarrow$ Algorithm seemed to "forget"

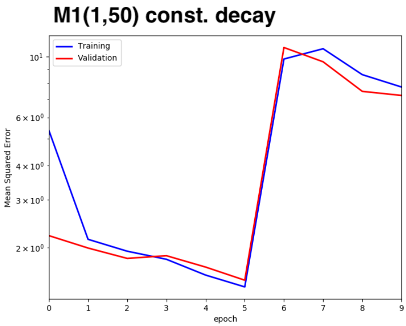
- Introduced regularization

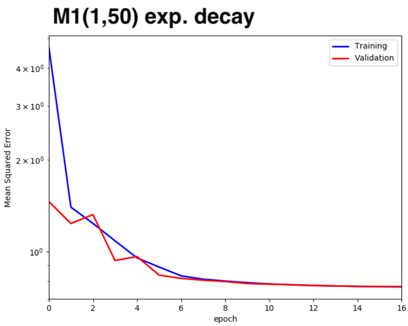
$\Rightarrow$ No significant effect on performance

- And many more...

Lesson(s) learned...

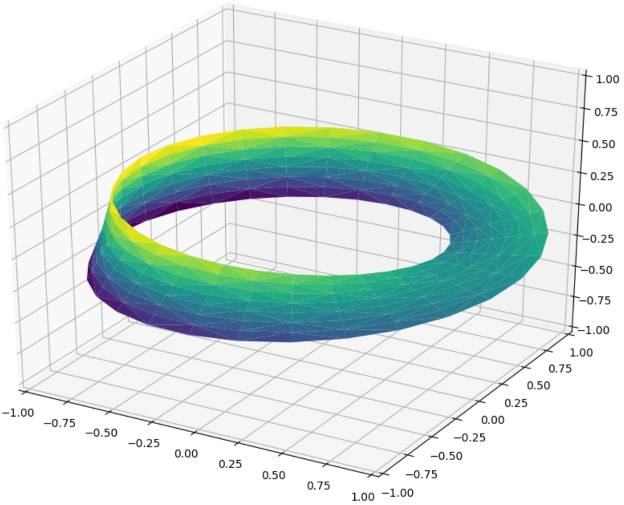
- Always check the training curve(s)





Lesson(s) learned...

- Always check the training curve(s)
- Always check the shape of your data, when using tensorflow

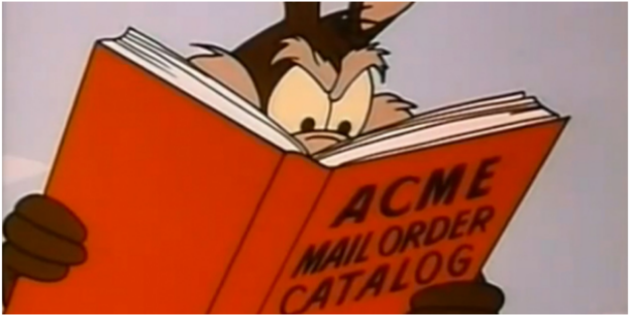


Lesson(s) learned...

- Always check the training curve(s)
- Always check the shape of your data, when using tensorflow
- Pre-training / transferring weights can be very helpful

Lesson(s) learned...

- Always check the training curve(s)
- Always check the shape of your data, when using tensorflow
- Pre-training / transferring weights can be very helpful
- Always read the manual $\Rightarrow$ Had to re-do significant amount of work



Picture taken from **here**

Lesson(s) learned...

- Always check the training curve(s)
- Always check the shape of your data, when using tensorflow
- Pre-training / transferring weights can be very helpful
- Always read the manual $\Rightarrow$ Had to re-do significant amount of work
- Think about how you train your algorithm
  $\Rightarrow$ Fixed lstm-parameters during the second stage training

Lesson(s) learned...

- Always check the training curve(s)
- Always check the shape of your data, when using tensorflow
- Pre-training / transferring weights can be very helpful
- Always read the manual $\Rightarrow$ Had to re-do significant amount of work
- Think about how you train your algorithm
  $\Rightarrow$ Fixed lstm-parameters during the second stage training
- Do not be afraid to push your model to the limits