

---

---

# D3 Visualizations

Atif Ahmed (aa3931)

---

---

# Outline

- Overview of the neural architecture
- Problem statement
- Preprocessing
- Prototypes/Approaches
- Separate visualizations
- Full visualization
- Technologies used
- What next?
- References

# Overview of the neural architecture

- Genomic and demographic data of large number of patients.
- High dimensional data (~4000 features).
- Uses group lasso and cox regression model for survival analysis.
  - Group lasso is used for taking care of any related features.

# Overview of the neural architecture

- Uses only 2 layers (1 hidden and 1 output) and generates pretty 0.95+ accuracy.

```
~/Downloads/Backup/Stud/MS/Sem2/Project/d3_vis — atifahmed@biocluster:~/project/deeplearningMentor/raw
_cnv RNU6ATAC35P_cnv RNU6ATAC37P_cnv RNU6ATAC40P_cnv RNU6ATAC41P_cnv RNU6ATAC5P_cnv RNU6ATAC_cnv RNU6ATAC_gen
P8_cnv RNY1_cnv RNY3P2_cnv RNY3P4_cnv RNY3P5_cnv RNY3P6_cnv RNY3P8_cnv RNY3_cnv
24_cnv RNY4P25_cnv RNY4P27_cnv RNY4P28_cnv RNY4P29_cnv RNY4P2_cnv RNY4P30_cnv RNY4P
RNY4_cnv RNY5P3_cnv RNY5P8_cnv RNY5_cnv ROBLD3_gene ROB01_gene ROB02_gene
O3_mut ROB04_cnv ROB04_gene ROB04_mut ROCK1_gene ROCK1_mut ROCK2_cnv ROC
ROGDI_mut ROM1_gene ROM1_mut ROM01_gene ROPN1B_gene ROPN1B_mut ROPN1L_gen
1_gene ROR1_mut ROR2_cnv ROR2_gene ROR2_mut RORA_gene RORA_mut ROR
ROS1_cnv ROS1_gene ROS1_muRP1-177G6.2_gene RP11-644F5.10_miRNA RP1L1_gene RP1L1_mu
P9_gene RP9_mut RPA1_gene RPA1_mut RPA2_cnv RPA2_gene RPA2_mut RPA3_gene R
AP1_cnv RPAP1_gene RPAP1_mut RPAP2_cnv RPAP2_gene RPAP3_gene RPAP3_mut RPE
RPF1_gene RPF1_mut RPF2_cnv RPF2_gene RPF2_mut RPGRIP1L_gene RPGRIP1L_mut
v RPH3AL_gene RPH3AL_mut RPH3A_cnv RPH3A_gene RPH3A_mut RPIA_gene RPIA_mut
L10L_mut RPL10_cnv RPL10_gene RPL11_cnv RPL11_gene RPL11_mut RPL12_gene RP
ne RPL13A_gene RPL13P5_gene RPL13_cnv RPL13_gene RPL13_mut RPL14_gene RPL14_mut
8A_gene RPL18_cnv RPL18_gene RPL19P12_gene RPL19_gene RPL21P122_cnv RPL21P44_gene RPL2
RPL22L1_mut RPL22P19_cnv RPL22_cnv RPL22_gene RPL22_mut RPL23AP32_gene RPL23AP53_cn
AP7_gene RPL23AP82_gene RPL23A_gene RPL23P8_gene RPL23_gene RPL24_gene RPL26L1_cnv RPL26
RPL27_cnv RPL27_gene RPL28_cnv RPL28_gene RPL28_mut RPL29P2_gene RPL29_cnv
0_gene RPL31P11_gene RPL31_gene RPL32P3_gene RPL32_gene RPL34_gene RPL35A_cnv RPL3
RPL36_gene RPL36_mut RPL37A_gene RPL37_cnv RPL37_gene RPL38_cnv RPL38_gene
L_gene RPL3L_mut RPL3_gene RPL3_mut RPL41_gene RPL4_gene RPL5_cnv RPL5_gene
mut RPL7A_cnv RPL7A_gene RPL7A_mut RPL7L1_gene RPL7_gene RPL8_gene RPL8_
RPLP0_gene RPLP0_mut RPLP1_gene RPLP2_cnv RPLP2_gene RPN1_gene RPN2_gene
21_mut RPP25_gene RPP30_gene RPP30_mut RPP38_gene RPP38_mut RPP40_cnv RP
RPRD1A_mut RPRD1B_gene RPRD1B_mut RPRD2_cnv RPRD2_gene RPRD2_mut RPRML_gene
```

# Problem Statement

- We need to improve accuracy and predicted probabilities by fine tuning the neural architecture.
- For fine tuning the architecture, different combinations of hyperparameters such as number of hidden units, number of iterations, the activation function used etc. should be tried.
- Therefore, we need to build a tool to visualize these various configurations and how the neural network is working at each iteration.

# Preprocessing

- Output produced by the model is stored in 4 different files:
  - test...predict.txt
  - test...groundTruth.txt
  - test...result.txt
  - test...weight.txt
- Simple non-serialized text files which makes getting object data from them impossible. Javascript cannot read it.

# Preprocessing

- We need to write our own scripts to convert the text files into a structured format (json).

- test...predict.txt

```
[ 1.60038117e-02]
[ 7.18286072e+02]
[ 1.19338656e+08]
[ 1.06018760e+04]
[ 2.78654102e+13]
[ 4.36446571e+00]
[ 3.24383319e+13]
[ 4.85082893e+01]
[ 9.08316461e+13]
[ 5.78567383e+03]
[ 1.10019975e+14]
[ 2.19703297e+14]
[ 4.73995557e+03]
[ 3.25056251e+15]
[ 1.03642120e+01]
```

```
{
  "step": 25,
  "probs": [
    {
      "index": 0,
      "prob": 0.00267549767
    },
    {
      "index": 1,
      "prob": 0.00463634264
    },
    {
      "index": 2,
      "prob": 0.00606542407
    },
    {
      "index": 3,
      "prob": 0.00825021509
    },
    {
      "index": 4,
      "prob": 0.0720237345
    }
  ]
}
```

# Prototypes/Approaches

- D3.js was identified as the best approach. Has been used by [playground.tensorflow.org](https://playground.tensorflow.org) also.
- First, existing designs were tried to visualize the network diagram of iCAGES.
- It was decided that predicted probabilities for patients should be shown with circles with colors density according to the magnitude of the probability.



# Prototypes/Approaches

- It was decided that the accuracy and cost at each step (iteration) should be updated and shown dynamically.
- For the network architecture, basic circles and other svg shapes can be used.

# Separate visualizations

- Visualizations were divided into 3 components:
  - Network architecture with changing output probabilities at each step.
  - Circles changing color densities according to the predicted probability at that step.
  - Dynamic visualization of accuracy and cost at each step.

# Full visualization

- Integrated and synchronized all separate visualizations which are updated now simultaneously at each step.
- Dropdown options to update various hyperparameters. Visualization can be started with new choice of hyperparameters.

# Technologies used

- d3.js for all the visualizations.
- Python for converting text data to d3-readable json files.
- SimpleHTTPServer for local web hosting and Github pages for hosting on Github.
- Twitter Bootstrap for prettifying the UI.

# What next?

- We need to be able to show weight changes on the architecture.
- We need to be able to change even more hyperparameters.
- Use something like convnet.js to make everything only on browser and specific to our dataset and network (e.g. [playground.tensorflow.org](https://playground.tensorflow.org)).

# References

- d3Vienno tutorials (20 videos) for d3.js  
<https://www.youtube.com/user/d3Vienno>
- Blocks blog for better understanding of d3.js transitions and updating with new data  
<http://bl.ocks.org/benjchristensen/1148374>
- D3.js official examples for initial prototyping  
<https://github.com/d3/d3/wiki/Gallery>